Neural Plausibility of Bayesian Inference

by

Sugandha Sharma

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master of Applied Science
in
Systems Design Engineering

Waterloo, Ontario, Canada, 2018

© Sugandha Sharma 2018

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

Behavioral studies have shown that humans account for uncertainty in a way that is nearly optimal in the Bayesian sense. Probabilistic models based on Bayes' theorem have been widely used for understanding human cognition, and have been applied to behaviors that range from perception and motor control to higher level reasoning and inference. However, whether the brain actually uses Bayesian reasoning or such reasoning is just an approximate description of human behavior is an open question. In this thesis, I aim to address this question by exploring the neural plausibility of Bayesian inference.

I first present a spiking neural model for learning priors (beliefs) from experiences of the natural world. Through this model, I address the question of how humans might be learning the priors needed for the inferences they make in their daily lives. I propose neural mechanisms for continuous learning and updating of priors - cognitive processes that are critical for many aspects of higher-level cognition. Next, I propose neural mechanisms for performing Bayesian inference by combining the learned prior with the likelihood that is based on the observed information. Through the process of building these models, I address the issue of representing probability distributions in neural populations by deploying an efficient neural coding scheme. I show how these representations can be used in meaningful ways to learn beliefs (priors) over time and to perform inference using those beliefs.

The final model is generalizable to various psychological tasks, and I show that it converges to the near optimal priors with very few training examples. The model is validated using a life span inference task, and the results from the model match human performance on this task better than an ideal Bayesian model due to the use of neuron tuning curves. This provides an initial step towards better understanding how Bayesian computations may be implemented in a biologically plausible neural network. Finally, I discuss the limitations

and suggest future work on both theoretical and experimental fronts.

Acknowledgements

I would like to thank

- **Chris Eliasmith**, for his supervision and support throughout the time I have been a part of the Computational Neuroscience Research Group, for the direction he provided to the ideas and proposals I had, and for his valuable feedback during the brainstorming research sessions.
- **Aaron Voelker**, for his insights on the algorithms derived in sections 4.3.1 and 4.3.2, and for his suggestion to use the plate notation in figure 4.2.
- **Terry Stewart**, for inspiring the work in this thesis through his presentation on function representations delivered to the Computational Neuroscience Research Group.
- **Bryan Tripp** and **Paul Marriott** for reviewing and providing feedback on this thesis.
- My **labmates** for their comments and fruitful discussions during my presentations over the development period of the work in this thesis.

I express my sincere gratitude to everyone of you for your guidance and support.

Dedication

This thesis is dedicated to my parents, Promila Sharma and Ravi Kant Sharma who are the most loving parents ever; my aunt and uncle, Anupma Sharma and Bal Krishan Sharma who have always loved me even more than their own children; my friend Marc Vaz who encouraged my decision to pursue what I was passionate about; my brother Dhruv Sharma and my cousins Shruti Sharma and Shivam Sharma for their constant love, support, encouragement and inspiration.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Human beings are able to understand and make sense of the information we perceive on a daily basis. But, how do we make sense of all the information we perceive? When we look at an object, how are we able to deduce all its attributes like color, shape etc.? One possible hypothesis is that we are able to do these things due to the knowledge bank which we have acquired from our prior experiences. For example, we are able to infer the sense of a word having different meanings based on the context it is being used in. This is because the context provides additional information that helps us pick the correct meaning of the word from our knowledge bank. However, what if the given word does not exist in our knowledge bank (i.e., we have never seen that word before)? In this case, we are still able to guess what the meaning of the word might be, based on the contextual information provided. This indicates that the human brain is not only able to recall what it had learned before, but also use that information to make inferences. We make such inferences on a daily basis e.g., trying to reason out the best move in a game of chess, inferring the personality of people we meet based on a short interaction with them, inferring the speed

of an approaching car when crossing a road etc.

In most of these cases, external stimuli interact with internal factors such as our emotions or memories to influence how we act. However, when the brain perceives the physical world to make a decision or to take an action, it needs to deal with the uncertainty associated with the sensory system as well as its own knowledge. Moreover, computations performed by the nervous system are also subject to uncertainty because of the influence of cellular and synaptic noise. Thus, at the level of cognition, the models that the brain uses to interact with its environment must necessarily cope with missing and imperfect information about the world. This suggests that being able to account for probabilistic contingencies in the environment and stochasticity within our own nervous system can be useful for cognitive functions like perception, decision making, thinking, planning, etc.

The Bayesian framework provides a coherent way of dealing with these uncertainties. Bayesian methods are based on the Bayes' theorem, which states that one's belief about the world should be updated according to the product of what one believed before and what evidence has been observed since. It offers a mathematically rigorous approach to combine the prior knowledge with the incoming evidence. The term "belief" is commonly used to describe one's knowledge of probabilistic information about variables that either describe the state of the world or describe a mental state. It can be considered as one's subjective probability implying that a person with a stronger belief in a given proposition has a higher subjective probability (Ma and Jazayeri, 2014). Additionally, uncertainty is usually specified in terms of the width of the belief distribution. For example, when trying to gauge the speed of an approaching taxi, one could maintain a belief distribution over the taxi speed. In this case, the sensory uncertainty can be defined as the standard deviation of the belief distribution over the taxi speed (e.g., 60 ± 5 km/h). In case of fog or at night time, the uncertainty may be higher (e.g., 60 ± 15 km/h). While some beliefs may be

innate, others could be acquired through life experience, or some may be a combination of both.

Behavioral studies have confirmed that humans often account for uncertainty in a way that is nearly optimal in the Bayesian sense (i.e., "Bayes optimal") (Ma et al., 2006). This implies that (1) neural circuits must, at least implicitly, represent probability distributions, and (2) neural circuits must be able to effectively compute with these probability distributions in order to perform Bayesian inference near-optimally. Probabilistic models based on Bayes' rule have been widely used for understanding human cognition including inference, parameter and structure learning (Jacobs and Kruschke, 2011), and word learning (Xu and Tenenbaum, 2007).

However, most Bayesian models lack biological plausibility because it is unclear how these computations might be realized in the brain. In particular, many of these models rely on sophisticated computations, including high-dimensional integration, precise multiplication, and large-scale structure representation. But, few demonstrate how such computations could be performed with spiking neurons. This is one of the questions I address in this thesis: How can we represent probability distributions in spiking neural circuits, and how can we use these representations in meaningful ways to perform inference.

A biologically plausible Bayesian approach can provide us insights into the working of the brain at multiple levels of analysis (Eliasmith, 2013). Moreover, it can also help in making more accurate normative predictions about how the perceptual system combines prior knowledge with sensory observations, enabling more accurate interpretations of data from psychological experiments (Doya, 2007). And finally, it can point the way towards approximate Bayesian algorithms that are efficiently implemented in a neural substrate.

Griffiths et al. (2012) conclude that different theoretical frameworks, such as Bayesian

modeling and connectionism, have different insights to offer about human cognition, distributed across different levels of analysis. In this thesis, I make an initial attempt towards integrating these levels. I explore the neural basis of Bayesian inference to answer questions about how beliefs (prior distributions) can be learned through life experience, how these distributions can be represented in a connectionist framework using spiking neurons, and how the neural representations of these distributions can be used in meaningful ways to reproduce human behavior.

1.1 Thesis Organization

In **chapter 2, background,** I provide an overview of the Bayes' theorem and set the context that it is used in within this thesis (i.e., cognitive modeling rather than data analysis and psychometric models). Given the focus on cognitive modeling, I describe different levels of analysis that the brain can be modeled/studied at, and provide motivation for bridging these levels in the context of Bayesian models of cognition. I end it with a review of different neural coding schemes for probability distributions.

In **chapter 3, computational neuroscience,** I describe the theoretical framework used to build the neural models presented in this thesis. Specifically, I present the Neural Engineering Framework (NEF) that describes how information can be represented in neural populations, how these representations can be transformed within neural networks and how their interactions can lead to dynamics in neural systems. Additionally, I also explain how the NEF can be used for neural coding of probability distributions using an explicit probability code. These methods are necessary to understand the models presented in subsequent chapters.

In **chapter 4, a spiking neural model for learning priors,** I detail the procedure

used to build a neural model spanning the levels of analysis presented in chapter 2. I ask the question: "How does the human brain learn the priors (beliefs) it uses for making daily life inferences?"; and propose a neural model to answer that question. The model is built based on the methods explained in chapter 3. The assumptions underlying the model are stated, its architecture is explained and the results showing the successful convergence of the model to the near optimal priors are presented.

In **chapter 5, a spiking neural model for Bayesian inference,** I extend the model presented in chapter 4 to include the neural circuitry for performing Bayesian inference. I illustrate (using the life span inference task) that the priors learned by the neural model can be successfully used to perform Bayesian inference in a neurally plausible way. The predictions made by the neural model are compared to human predictions on the life span inference task and results showing a close match are presented.

In **chapter 6, discussion and conclusions,** I address some of the unaddressed topics regarding the neural model presented in chapter 5 and propose directions for future research. Finally, I conclude the thesis by revisiting the goals listed in the next section.

1.2 Thesis Goals

The purpose of this thesis is to propose low-level neural realizations for the high-level Bayesian computations commonly used in models of cognitive inference. These models generally use a combination of a prior (previous knowledge) and a likelihood (derived from new observations) to make inferences in a given task. Proposing neural realizations for such computations is quite challenging since it involves answering several questions, including: How do humans acquire the knowledge that they use for making daily life inferences? How much of this knowledge is innate and how much is learned? How does the brain learn it

from experiences of the natural world?. Given how and what prior knowledge is acquired, additional questions are raised, such as: How is this knowledge represented in our brain? How is it used for performing inferences in our daily lives? How are the computations to do so realized in the brain?.

A complete story answering all of these questions would be overly ambitious, so I make following the goals of this thesis:

- To illustrate a systematic approach for integrating the different levels of analysis i.e., from high-level abstract computations to low-level neural mechanisms through the process of building a neurally plausible Bayesian model;
- To propose neural mechanisms that may be used for learning priors from experiences of the natural world, based on clearly defined assumptions;
- To show how these priors (and any probability distribution in general) can be represented in neural circuits using an efficient neural coding scheme;
- To propose neural mechanisms for combining neural representations of learned priors with those of new observations (i.e, likelihood) to perform Bayesian inference; and
- To reproduce human behavioral results for an inference task (the life span inference task) using the neurally plausible Bayesian model of cognitive inference that is proposed in this thesis.

Chapter 2

Background

2.1 Introduction

Bayes' theorem lies at the heart of the Bayesian inference. It was first proposed by a mathematician Thomas Bayes in the 18th century. The theorem provides a way to revise existing beliefs or predictions given new additional evidence which has not been observed before. Bayes' theorem has widespread applications ranging from financial analysis (e.g., for predicting stock prices), to medical diagnosis (e.g., to determine the accuracy of a medical test), to spam filtering, and any application where predictions need to be made by combining previous knowledge about a variable quantity with new information. Though the Bayesian framework has been expanded to include other methods for approximate inference e.g., variational Bayesian methods, Monte Carlo sampling methods, etc., understanding of such methods is out of scope.

In this chapter, I start by providing a review of the Bayes' theorem and explain the different contexts that it is most commonly used in. I draw distinctions between the three

kinds of Bayesian models: cognitive, data analysis and psychometric, and clarify that it will be used in the cognitive modeling context in this thesis. This is followed by a review of the different levels of analysis at which Bayesian models of cognition can be built to study the brain. The interdependence of these levels is emphasized, and a recent extension of the levels is described. Motivation for bridging these levels of analysis specifically for Bayesian models of cognition is provided through a discussion of the significance of Bayesian cognitive models. As an initial step in this direction, different theoretical schemes for encoding probability distributions in neurons and neural populations are reviewed.

2.2 Bayes' Theorem

Bayes' theorem is an elementary result of probability theory that Bayesian inference grew out of. Given two random variables, H and D, probability theory can be used to write the joint probability, $p(H, D)$, of these two variables as follows:

$$p(H, D) = p(H|D)p(D) = p(D|H)p(H) \tag{2.1}$$

This implies:

$$p(H|D) = p(D|H)p(H)/p(D) \tag{2.2}$$

The above expression provides a method to compute the conditional probability of H given D from the conditional probability of D given H and is known as Bayes' theorem.

If we assume that D represents some observed data, then Bayes' theorem can be used to infer the process responsible for generating that data. Let H represent the set of hypotheses about this process, and let $p(H)$ indicate one's degree of belief in those hypotheses, i.e.,

the probability that a given H is the true generating process (also known as the prior probability). Bayes' theorem provides a way to update one's beliefs about various hypotheses in the light of the evidence provided by the data D, by computing the posterior probability, $P(H|D)$, i.e., the degree of belief in the hypothesis H conditioned on the observed data D (given by equation 2.2).

Bayes's theorem and related methods have been used to study the brain by building Bayesian models in mainly three alternative contexts:

1. Data Analysis: Bayesian data analysis methods (Friedman et al. 2000: part 5, Kruschke 2014) have been widely used in analyzing experimental data in various fields including neuroscience. In this case, the associated Bayesian model and its parameters describe the trends in the data obtained empirically, e.g, neuro-imaging data, expression data for thousands of genes, etc. (Stephan et al. 2009, Trujillo-Barreto et al. 2004, Penny et al. 2011). However, the model parameters do not refer to anything in the process that generated the data.

   Models that fall into this category usually contain a prior on some parameters, and use the experimental data to generate a posterior (on these parameters) that explains the experimental data. These models do not make any claims about the perceptual, cognitive or neural processes that may have lead to the data being analyzed.

2. Psychometric Models: Bayesian models whose parameters describe trends in behavioral data fall into this category (Levy and Mislevy 2016, Kuss et al. 2005). Though the parameters in these models refer to mental constructs, the models provide a computational level explanation, and there doesn't need to be a Bayesian process that generated the data. Models in this category are similar to the data analysis models, except for the fact that the data is generated by the "mind" of the subjects

involved in the behavioral study. These models provide explanations of behavioral data without making any claim about the Bayesian processes occurring in the brain of the subjects.

3. Cognitive Models: These are the models that aim at explaining human cognitive processes like reasoning, perceptual inference, causal learning, language processing, etc.(Griffiths et al. 2008, Chater et al. 2010). These models are of the view that the Bayesian processes occur within the brain of the subject. The parameters of these models describe mental representations, and the Bayesian inference is done by the "mind" that given some observed data about the world, tries to draw conclusions about its underlying structure, and then uses this knowledge to make inferences about new cases.

   The knowledge is represented in the form of a prior on parameters and the observed data is usually the sensory data perceived by a subject. These models use the prior and the sensory data to generate a posterior on parameters, that explains the sensory data. All components of the model are thought to exist within the subject's brain, except the sensory data, which comes from the outside world.

While all of the above models are important applications of Bayes' theorem, in this thesis, I focus on using Bayes' theorem for building Cognitive Models.

In cognitive modeling, Bayes' theorem provides a method to combine the sensory information with one's prior beliefs in an optimal way. Bayesian inference about a variable Z e.g., the speed on an approaching taxi, based on sensory information X is performed by computing the posterior probability $p(Z|X)$, which specifies what is known about the variable Z after the sensory information has been perceived. Bayes' theorem states that the posterior probability can be computed as follows:

$$p(Z|X) = p(X|Z)\ p(Z)\ /\ p(X) \tag{2.3}$$

Here p(Z) is the prior that describes the probability distribution of Z before the perception of any sensory information. In other words, it captures one's prior beliefs about the variable Z. The probability $p(X|Z)$ is a function of Z and is commonly called the likelihood function. It describes how the sensory information depends on the variable Z.

The posterior probability $p(X|Z)$ quantifies how much one should believe a particular Z, given the sensory information observed. The probability $p(X)$ is the sum of probabilities of the sensory information arising under all possible values of Z, and is given by $p(X) = \sum_i p(Z_i)\ p(X|Z_i)$. Thus, the posterior probability $p(Z_i|X)$, of a particular value Z_i of variable Z, is the ratio of the probability of the sensory evidence under Z_i (the product of its prior and likelihood: $p(X|Z_i)\ p(Z_i)$) relative to the total probability of the sensory evidence arising under all possible values of the variable Z (the sum of the prior-likelihood products for all possible values of Z i.e., $p(X)$).

2.3 Levels of Analysis

Bayesian framework can be used to build models of cognition at different levels of analysis as originally proposed by David Marr (Marr, 1982) and extended by Tomaso Poggio (Poggio, 2012). Marr proposed that complex information processing systems like the brain be should be studied at different levels as given below:

- Computational level: At the computational level, the Bayesian framework can be used to specify the nature of the problems that the brain faces, i.e., what are the

goals, what computations might be appropriate to achieve these goals, what is the structure of the environment these goals should be achieved in, and what is the logic of the strategy to achieve them.

- Algorithmic level: This forms a bridge between computational and implementation levels, and answers questions on how the computational theory proposed at the computational level is implemented, i.e., how are the inputs and outputs represented, what computational operations can be employed over those representations, what algorithms are used for the necessary transformations, etc. Work done at the computational level might tell us that the brain faces probabilistic challenges, however our brain might solve them using some approximations, e.g., to save resources. Such computations may also be subject to the brain's representational and architectural constraints.

- Implementation level: This level states how the representations and algorithms proposed at the algorithmic level can be realized in the physical substrate, e.g., biologically in the brain through the wetware, cells and circuits. It involves mapping the algorithmic Bayesian formalism to neuronal processes. For example, it is possible that that individual neurons or populations of neurons may directly encode probability distributions, and that some neural processes might be carrying out probabilistic inference (Ma et al. 2006, Narain et al. 2018).

Though some people might be of the view that explanations at different levels above are largely independent of each other (e.g., a software engineer doesn't need to know the computer hardware in detail), Poggio (2012) has argued that these levels are highly interdependent since the knowledge at the implementation level provides constraints for the algorithmic level, and contributes to the characterization of the problem at the compu-

tational level. Additionally, insights gained at the computational and algorithmic levels help in asking the right questions and performing the right kind of experiments at the implementation level. Besides emphasizing the connections between these levels, Poggio has also extended the range of levels to include evolution and learning. In his own words: "not only we need to know how transistors and synapses work, which algorithms are used for computations such as stereo, and what are the goals and the constraints of computations such as object recognition, but we also need to understand how a child may learn recognition and how evolution could have discovered the learning algorithms themselves. In short, in addition to the old, classical levels, we also need to understand what is the role of nature and nurture in the development of intelligence".

The two new levels added by Poggio are described below:

- Learning: Poggio has proposed that the level of learning should be added to the list of levels of understanding above the computational level. This level asks questions including: How are the representations in the brain learned? How do humans learn from experiences of natural world? How do humans learn from very few examples? What are the learning algorithms employed by humans and What are their a priori assumptions?

- Evolution: This level was added above the learning level and suggests that the effectiveness with which humans learn and perform in the world is partly due to the huge bank of prior knowledge. Questions asked at this level are: How do the computations in the brain evolve with evolution? How is evolution responsible for intelligence? What is the path taken by evolution to develop intelligence? How does learning evolve by itself in a given species?

Note that the two levels added by Poggio do not seem to be the same kinds of levels as

proposed by Marr, i.e., they are not abstractions. For instance, we can think of learning as having a computational, algorithmic and implementation level [1]. However, through these two levels, Poggio underscores the importance of understanding how an organism (or a species as a whole) is able to learn and evolve the computations and representations that allow it to survive in the natural world.

2.4 Bridging levels for Bayesian models of cognition

Many behavioral studies and psychophysical experiments suggest that the human brain represents and manipulates uncertain information in terms of probability distributions in some way. As a result, probabilistic models based on Bayes' rule have been widely used to answer questions about how the brain makes generalizations and predictions about the world, using the noisy and incomplete information available to it. Some of the domains to which this framework has been applied include language learning (Xu and Tenenbaum 2007, Goldwater and Griffiths 2007, Teh 2006, Griffiths and Kalish 2005), motor control (Wolpert and Landy, 2012), visual perception (Mamassian et al. 2002,), inductive inference (Tenenbaum et al. 2006, Brody and Lapata 2009), and causal learning (Gopnik et al. 2004, Waldmann and Martignon 1998, Lu et al. 2008).

However, most of these models are built at the computational level, i.e., they provide an ideal solution to an abstract statistical problem that people must solve, without identifying the cognitive or neural processes involved in representing and transforming information,

[1] Keith Nishihara expressed using a diagram presented in the appendix of Poggio (2012), that it may be more useful to think of learning as having a computational, algorithmic and implementation level.

and without addressing the question of how such processes might be realized in the brain. In order to fully assess the implications of the Bayesian framework for improving our understanding of cognition and the brain, it is important for it to bridge the different levels of analysis discussed in section 2.3.

One of the goals of this thesis is to bridge these levels of analysis by proposing neural mechanisms for Bayesian computations. Since Bayes' rule is based on probability theory, an obvious first question to ask is: How are probability distributions represented in the brain? The study of probabilistic computations by the brain is still in its early stages, therefore which neural code or set of codes the brain uses is an open question. I discuss some of the common proposals for neural coding of probability distributions in the next section.

2.5 Neural Coding of Probability

Neural activity is generally thought to encode low-dimensional values. For instance, neurons in the middle temporal (MT) visual area are interpreted as encoding the direction of motion while the neurons in the primary visual cortex (V1) are interpreted as encoding orientation. Additionally, vector representations have also been found in a wide variety of neural systems. For instance, neurons in the monkey motor cortex have been found to represent arm movements using a weighted vector code. Moreover, besides direction of motion, neurons in MT have been found to simultaneously encode speed, contrast, disparity, attention, and spatial extent. Similarly, besides orientation, neurons in V1 simultaneously encode spatial frequency and disparity. There are also large contextual influences in each case, which bring in more dimensions (corresponding to the other parts of the visual space).

Over the span of last decade or two, several groups have proposed that neural activity

might also encode functions of variables, i.e., probability distributions that are high (infinite) dimensional. This indicates that neural computations must be able to manipulate these distributions according to the rules of probabilistic inference, while at the same time staying faithful to the constraints imposed by the neural substrate, i.e., the brain. In order to understand how such computations are possible at the neural level, we first need to discuss the neural codes. Several groups have proposed theoretical schemes for neural coding of probability distributions. I review the prominent ones in this section.

2.5.1 Probabilistic population codes

Probabilistic population codes describe one possible way of how the brain might use population activities for probabilistic computations. Proponents of this kind of code claim that instead of encoding the value of a stimulus, the firing pattern of a neural population encodes a probability distribution, $p(R|X)$ over the possible values of a stimulus X, where R is the firing rate of the neural population (Ma et al. 2006, Jazayeri and Movshon 2006, Shi and Griffiths 2009). Note that $p(R|X)$ can be seen as the likelihood of stimulus X, that reflects the probability of the firing pattern given the stimulus X. The stimulus represented by the neural population can thus be recovered through Bayesian decoding. Under this code, neural operations performed on the neural populations correspond to manipulations of the corresponding likelihood functions, and uncertainty is implicitly represented in the population activity.

In variants of this coding scheme, the likelihood over the stimuli is proportional to the activity of the neural population (instead of being directly encoded in the neural activity). For instance, Jazayeri and Movshon (2006) propose that the contribution of each neuron in the neural population to the log of the likelihood function encoded by the population,

is given by the product of the neuron's response and the logarithm of its tuning curve.

To use this code for making predictions, one needs to assume some form of neural variability $p(R|X)$, e.g, Poisson-like variability is one of the commonly used family of distributions. Fetsch et al. (2012) trained monkeys to perform a head discrimination task from visual and vestibular cues, randomly varying cue reliability. Assuming a Poisson-like probabilistic population code, they decoded likelihoods from neural responses in the dorsal medial superior temporal (MSTd) cortex. This is the area where visual and vestibular cues for self-motion are integrated. They were able to accurately predict the monkey's cue integration behavior with a moderate deviation from the optimal prediction.

2.5.2 Sampling codes

The sampling codes assume that the probability of a variable of interest is directly mapped on to the firing rate of neurons. Under this code, neurons correspond to variables, and their firing encodes the distribution of the variables. At a given point in time, the activity of a neuron is a sample from the distribution that needs to be represented (Hoyer and Hyvärinen 2003, Moreno-Bote et al. 2011, Fiser et al. 2010). For instance, to encode a distribution $p(Z|X)$, a neuron can be used corresponding to variable Z such that it has firing rates that are samples from the distribution $p(Z|X)$. A higher mean firing rate of the neuron would imply that a large value of variable Z is most probable, while a lower mean firing rate would imply that a small value of Z is most probable. A higher variability in the firing rate over time represents a higher uncertainty.

The sampling hypothesis leads to the prediction that the distribution of spontaneous activity of a neuron might reflect the statistics of the environment. This prediction is consistent with the finding of Berkes et al. (2011) in the ferret visual cortex where the

neural activity was found to reflect the environmental statistics. However, sampling codes also predict that the firing rate variability of the neuron encoding a distribution should increase with uncertainty. This is inconsistent with the finding of Tolhurst et al. (1983) who show that decreasing the contrast of a sinusoidal grating stimulus (increased uncertainty) leads to a decrease in the firing rate variability of neurons in the visual cortex.

2.5.3 Explicit probability codes

Under this hypothesis, a probability distribution is directly encoded by the neural activity of a population of neurons. The activity of a neuron tuned to a particular stimulus feature is monotonically related to the probability density of that feature usually through a linear or logarithmic transformation. This implies that the pattern of activity across the population of neurons matches the shape of the probability distribution that the population is encoding. For instance, to encode a distribution $p(Z|X)$, a population of neurons can be used such that each neuron has a preferred stimulus Z_i. The firing rate of a neuron with a preferred stimuli Z_i reflects the value of the distribution at Z_i. Neurons with preferred stimuli Z_i that have a lower probability have lower firing rates, while the neurons with preferred stimuli Z_i that have a higher probability have higher firing rates. A wider activation pattern across the population represents a higher uncertainty.

This coding scheme and its variants have been proposed and used by several groups (Barlow 1969, Barlow and Levick 1969, Anastasio et al. 2000, Eliasmith and Anderson 2003, Anderson and Van Essen 1994, Zemel et al. 1998, Sahani and Dayan 2003). For example, Barlow (1969) proposed a log probability code where the response of a neuron tuned to a particular image feature (feature detector neuron), is proportional to the log of the probability that their trigger feature is present. On the other hand, Anastasio

et al. (2000) proposed that neuronal responses of deep superior colliculus neurons are proportional to the probability that a target (source of stimulation) is present rather than to its log probability.

Furthermore, other researchers have proposed techniques to take advantage of the fact that probability distributions are functions. For instance, Eliasmith and Anderson 2003 have proposed that functions can be expressed as a sum of other functions called the basis functions. Through this approach, a probability distribution can be expressed as a linear combination of basis functions using a set of coefficients. Hence, the distribution can be represented by encoding this set of coefficients in the neural activity. One of the advantages of using this approach is that it requires lower number of resources (e.g., fewer neurons) relative to other explicit coding proposals, for representing the same distribution. This is because through this approach, only the coefficients need to be encoded rather than the entire original distribution.

2.6 Discussion

We have seen that Bayesian models of cognition have been used to understand a wide range of cognitive phenomenon that involve uncertainty. These range from visual perception and motor control to high-level reasoning and inference. This wide range suggests that Bayesian models may provide a link between the high-level and low-level cognition, thus bridging across the levels of analysis described in section 2.3.

Some neural coding schemes have already been proposed in an attempt to bridge the gap between abstract high level Bayesian models and their neural realizations (section 2.5). However, the study of Bayesian computations in the brain is still in its early stages and many open questions remain unanswered. For instance, an important open question

is whether probabilistic neural computations in the brain are optimal or suboptimal? In the context of Bayesian computations, suboptimality can be caused by neural networks implementing approximate computations. For instance in section 2.5.3, we saw that high dimensional distributions can be approximated by computing basis functions. Another cause of suboptimality might be incompletely learned beliefs (priors). For instance, during development, children learn about the world through their interaction with it. Their beliefs about the world change as they learn, and hence the inferences that they make might be suboptimal since their beliefs may not match the reality of the world. Another open question is: What are the learning mechanisms that enable the brain to update these beliefs? Though we know from psychological studies that humans have different types of beliefs about different phenomena in the world, it is unclear whether these beliefs are represented differently and whether it is neurally plausible to integrate different beliefs and to use the learned beliefs in analogous situations.

These are just a small subset of the unanswered questions regarding probabilistic computations in the brain. Clearly, additional theoretical and experimental work needs to be done to answer these question. In this thesis, I focus on the theoretical side and present a neural model to propose answers to some of these questions, with the goal of exploring the computational and neural underpinnings of human inference. Though I compare the model results to human behavioral data, conducting experiments to explicitly test the theoretical schemes proposed is left for future work. In the next chapter, I explain the framework used for building the said neural model, and show how explicit probability codes can be implemented to represent probability distributions in spiking neural populations within this framework.

Chapter 3

Theoretical Neuroscience

3.1 Introduction

Computational Neuroscience is the study of brain function in terms of information processing properties of the structures (neurons, synapses, circuits) of which it is composed. Researchers in this field use quantitative tools to understand what computations are performed in the brain, and how these computations give rise to behavior.

Traditionally, the field of computational neuroscience has worked at the Implementation level. It has taken a bottom-up approach, starting with the study of single neural cells, followed by small and large networks of cells through detailed mathematical models of single cells and small networks. On the contrary, the field of cognitive science has traditionally worked at the computational level. It has taken a top-down approach, starting with the study of behavior of living beings followed by an attempt to understanding how the brain facilitates these behaviors. Given these two different approaches to understanding the brain, it is important to bridge these levels of analysis (as explained in sections 2.3, 2.4).

One of the goals of this thesis is to bridge the gap between computational neuroscience and cognitive science through the development of neural models that can reproduce human behavior.

In this chapter, I provide a summary of the computational methods used to build such neural models. These models are used to frame hypotheses that can be directly tested by current and future biological and psychological experiments. Specifically, the models presented in this thesis were built using the methods of the Neural Engineering Framework (NEF; Eliasmith and Anderson, 2003). The NEF specifies a method to implement a wide variety of functions using biologically plausible neuron models, and can be used to translate a mathematical description of a dynamical system to a spiking neural model (Eliasmith 2005, Singh and Eliasmith 2006). Thus, the NEF provides a well-defined way to explore how biologically plausible neural systems can implement dynamical functions. The NEF has been used to build a wide variety of neural models, including the current largest functional brain model (Eliasmith et al., 2012). Other neural systems modeled using the NEF include the spinal circuits in lamprey for controlling swimming, parts of the inner ear and brain stem for controlling a vestibulo-occular reflex, horizontal eye position control, directed arm movements and working memory (Eliasmith and Anderson 2003, Eliasmith 2013).

Importantly, in this chapter, I also detail how the NEF can be used for neural coding of probability distributions using an explicit probability code. An efficient coding scheme is proposed through the use of basis functions, which greatly reduces the neural resources required to represent the high-dimensional distributions by compressing them to a low-dimensional space. These low dimensional distributions are directly encoded by the neural activity of a population of neurons.

3.2 The Neural Engineering Framework

How do neurobiological systems represent the world? How do they use those representations through transformations to guide behavior? These are some of the central questions facing computational neuroscientists today. The Neural Engineering Framework (NEF) provides a neuroscientifically respectable approach to building biologically constrained models of neurobiological systems using the quantitative tools of engineering (e.g., dynamical systems theory, statistical inference, signal processing etc.; Eliasmith and Anderson, 2003). More precisely, it is a mathematical theory that provides methods for implementing algorithms on vector spaces in spiking neural networks, and through those methods, describes how biologically plausible neural systems can implement a wide variety of dynamical functions.

The NEF is based on three principles—representation, transformation, and dynamics—which specify how information can be represented in neural populations, how these representations can be transformed to compute functions, and how the interactions between these representations gives rise to dynamics in neural systems respectively. These principles are described in detail in the following sections.

3.2.1 Principle 1 – Representation

In order to characterize representation in a neural system, we need to specify encoding and decoding procedures. The principle of representation specifies a way of representing information in populations of neurons by identifying encoding and decoding strategies. It states that the activity of a neural population "represents" information in an underlying vector space. This information is represented as time-varying vectors of real numbers by

populations of neurons through nonlinear encoding (defined by the neuron tuning curves) and weighted linear decoding over the neural populations.

In general, a population of neurons has activities $a_i(\mathbf{x})$, which encode an n-dimensional stimulus vector, $\mathbf{x} = [x_1, x_2, \ldots, x_n]$, through the encoding given by:

$$a_i(\mathbf{x}) = G_i\left[J_i(\mathbf{x})\right], \tag{3.1}$$

where $a_i(\mathbf{x})$ are the activities of population of neurons, $G_i\left[\cdot\right]$ is the nonlinear transfer function describing the neuron's spiking response, and $J_i(\mathbf{x})$ is the current entering the soma of the neuron. The response function $G_i\left[\cdot\right]$ is determined by the intrinsic properties of the neuron e.g., absolute refractory period, resistances, capacitances, etc. that are typically used to model single neuron behavior. While various neural models with their own characterization of $G_i\left[\cdot\right]$ can be used to model single neuron behavior in the NEF, I have chosen $G_i\left[\cdot\right]$ to be the leaky integrate-and-fire (LIF) neuron model for purpose of the models presented in this thesis. The soma current is defined by:

$$J_i(\mathbf{x}) = \rho_i \left\langle \mathbf{e}_i, \mathbf{x} \right\rangle_n + J_i^{bias}, \tag{3.2}$$

where $J_i(\mathbf{x})$ is the current in the soma, ρ_i is a gain and conversion factor, $\mathbf{x}$ is the stimulus vector to be encoded, $\mathbf{e}_i$ is the encoding vector that corresponds to the "preferred stimulus" of the neuron—consistent with the standard idea of a preferred direction vector Schwartz et al. (1988)—and J_i^{bias} is a bias current that accounts for background activity. The notation $\langle \cdot, \cdot \rangle_n$ indicates an n-dimensional dot-product. The current $J_i(\mathbf{x})$ is used as the input to the LIF neuron model. The encoder, gain, and bias are randomly chosen, so different neurons have different background firing rates and different maximum firing rates along their preferred direction. This adds heterogeneity to the models by ensuring that

different neurons respond differently to the same stimulus. Neurons in a population also have different encoding vectors ($\mathbf{e}_i$) which implies that each neuron will fire most strongly for a particular combination of values of stimuli. Thus the tuning curve of a neuron is a function of the gain of the neuron (how quickly its activity rises), the bias (activity of a neuron given no input), and the encoder (the direction in the input vector space that the neuron is most responsive to).

Once the stimulus has been encoded by a population of neurons, it also needs to be decoded in order to characterize the computations performable by the neural network. With enough neurons, the originally encoded stimulus vector can be estimated from the activities of the population of neurons through a decoding process given by:

$$\hat{\mathbf{x}} = \sum_i a_i(\mathbf{x})\mathbf{d}_i. \tag{3.3}$$

The decoding vectors $\mathbf{d}_i$ (also known as "representational decoders") are typically found in the NEF by least-squares optimization, which I use in the models in this thesis (Eliasmith and Anderson, 2003). The decoders resulting from this optimization complete the definition of a *population code* over a set of neurons i for the representation of stimulus $\mathbf{x}$. As mentioned before, the code is defined by the combination of nonlinear encoding in equation 3.1 and weighted linear decoding in equation 3.3. To summarize, NEF defines neural representations as nonlinear encodings of vector spaces that can be linearly decoded.

Temporal representation

The population code does not explicitly address the issue of how information is encoded over time. To do so, we can begin by considering the temporal code for each neuron in isolation by taking the neural activities to be filtered spike trains as shown below:

$$a_i(t) = \sum_m h_i(t) * \delta(t - t_m) = \sum_m h_i(t - t_{i,m}), \tag{3.4}$$

where $\delta_i(\cdot)$ are the spikes at time t_m for a given neuron i, generated by $G_i[\cdot]$ and $h_i(t)$ are the linear decoding filters. We can compute the optimal filters for decoding using the NEF, however to make my models biologically plausible, I have chosen these filters ($h_i(t)$) to be the postsynaptic currents (PSCs) induced in subsequent neurons by the arrival of a spike. Eliasmith and Anderson (2003) have shown that this assumption causes minimal information loss, which can be further reduced by increasing the population size.

This temporal code can be combined with the population code defined before (equations 3.1, 3.2, 3.3), to provide a general *population temporal code* for vectors. The encoding and decoding equations for such a code are given by:

$$\delta(t - t_{im}) = G_i\left[\rho_i \langle \mathbf{e}_i, \mathbf{x} \rangle_n + J_i^{bias}\right] \tag{3.5}$$

$$\hat{\mathbf{x}} = \sum_{i,m} h_i(t - t_{i,m})\mathbf{d}_i. \tag{3.6}$$

3.2.2 Principle 2 – Transformation

Transformations of neural representations are functions of vector variables represented by neural populations. The principle of transformation describes a way to compute functions on neural representations in a biologically plausible way. Neurons communicate through synapses, and when a neuron spikes, it releases neurotransmitters across the synapse that causes some current to be imparted to the post-synaptic neuron. The NEF summarizes the many factors affecting the amplitude of the imparted current through a scalar connection weight representing the strength of the connection between the two neurons. To compute

any function on a connection, we set the connection weights between the two populations to be the product of the decoders in the first population and the encoders of the second population. The decoders can be determined based on the function we wish to compute.

For example, to compute a transformation $f(\mathbf{x})$ when two neural populations are connected, instead of finding the representational decoders $\mathbf{d}_i$ to extract the originally encoded variable $\mathbf{x}$, we can re-weight the decoding to specify some function $f(\mathbf{x})$ other than identity. In other words, we can find the decoders $\mathbf{d}_i^{f(\mathbf{x})}$ (also known as "transformational decoders"). In a communication channel, representational decoders $\mathbf{d}_i$ are computed by the least squares optimization $\langle ||\hat{\mathbf{x}} - \mathbf{x}|| \rangle_x$. To perform a transformation $f(\mathbf{x})$, instead of finding the representational decoders we can find the transformational decoders $\mathbf{d}_i^{f(\mathbf{x})}$ by using least-squares optimization to minimize the difference between the decoded estimate of $f(\mathbf{x})$ and the actual $f(\mathbf{x})$ given by $\langle ||\hat{\mathbf{x}} - f(\mathbf{x})|| \rangle_x$. This results in the transformation:

$$\hat{\mathbf{x}} = \sum_i a_i(\mathbf{x}) \mathbf{d}_i^{f(\mathbf{x})}. \tag{3.7}$$

Both linear and nonlinear functions of the encoded vector variable can be computed in this manner (Eliasmith and Anderson, 2003).

Encoders/Decoders v.s. Weights in Standard Artificial Neural Networks

In the NEF, when two neural populations having N and M neurons and D dimensions each are connected, instead of an $N \times M$ connection weight matrix, we have an $N \times D$ decoder matrix and a $D \times M$ encoder matrix. This provides for a huge speedup to the neural networks built using NEF as compared to the standard artificial neural networks which solve for an $N \times M$ connection weight matrix, since solving for these smaller matrices is relatively easier (as long as $N >> D$ and $M >> D$).

The equivalent synaptic weight matrix for computing the function $f(x)$ in the NEF is given by $\omega_{ij} = \rho_j \mathbf{e}_j \mathbf{d}_i^{f(\mathbf{x})}$, where i indexes the presynaptic population, j indexes the postsynaptic population, and $\mathbf{d}_i^{f(\mathbf{x})}$ are representational or transformational decoders. Thus, the connection weights in the NEF are determined by both the decoding of outgoing signals from the presynaptic population and the encoding of incoming signals in the postsynaptic population. This implies that changing a connection weight will change the transformation being computed as well as the tuning curve of the receiving neuron in the postsynaptic population. This is exactly what is expected to happen as is well known from work in artificial intelligence and computational neuroscience. The encoding/decoding distinction is made in the NEF because it is useful for understanding the representations in, and the functions performed by, the neurobiological system. However, neurobiological systems do not necessarily need to respect this distinction in order to perform their functions.

3.2.3 Principle 3 - Dynamics

Though feedforward transformations are sufficient to describe some neural systems, many neural systems require persistent activity through recurrent connections. The principle of Dynamics states that when recurrent connections are introduced, the values represented by neural populations can be considered as state variables in a dynamical system. The equations governing the dynamics of such neurobiological systems can be designed and analyzed using control theory methods, and translated into neural circuitry using the principles of Representation and Transformation explained above.

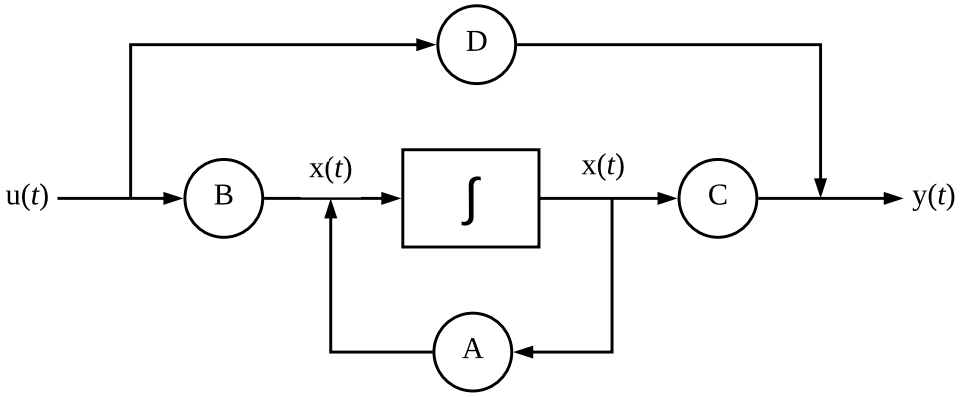


Figure 3.1: Block diagram of a linear time invariant system showing the input vector $\mathbf{u}(t)$, the state vector $\mathbf{x}(t)$, and the output vector $\mathbf{y}(t)$. $\mathbf{B}$ is the input matrix, $\mathbf{A}$ is the dynamics matrix, $\mathbf{D}$ is the feedthrough matrix and $\mathbf{C}$ is the output matrix. Figure adapted from Eliasmith and Anderson (2003).

In general, NEF provides a method for directly converting any set of differential equations into a biologically plausible spiking network approximation of the dynamics described by those equations (Eliasmith and Anderson, 2003). In this section, I explain how such a mapping can be obtained through the NEF.

We begin with the state equations that summarize the linear control theory:

$$\begin{aligned} \hat{\mathbf{x}}(t) &= \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \\ \hat{\mathbf{y}}(t) &= \mathbf{C}\mathbf{x}(t) + \mathbf{D}\mathbf{u}(t) \end{aligned} \tag{3.8}$$

Here (refer to figure 3.1), vector $\mathbf{x}(t)$ is called the *state vector*. It contains variables that collectively describe the internal state of the system, and summarizes the effect of all past input. All future output depends only on the value of the state vector and the future

input. Furthermore, $\mathbf{u}(t)$ is the input vector, $\mathbf{y}(t)$ is the output vector, $\mathbf{A}$ is the dynamics matrix, $\mathbf{B}$ is the input matrix, $\mathbf{C}$ is the output matrix, and $\mathbf{D}$ is the feedthrough matrix.

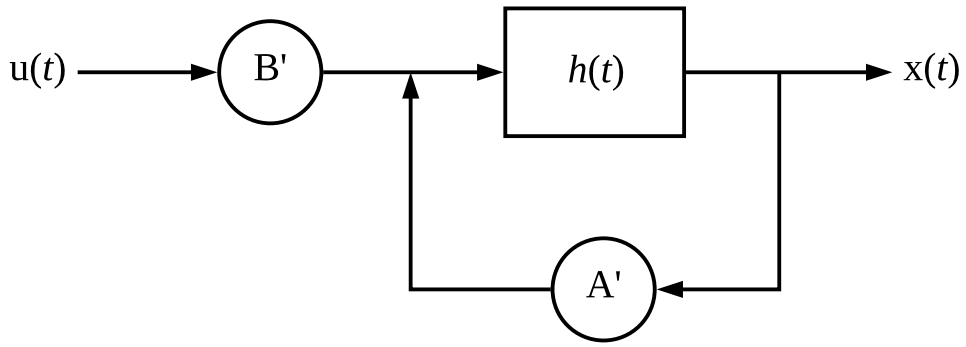


Figure 3.2: Block diagram of a neural population implementing a linear system, showing the input vector $\mathbf{u}(t)$, the state vector $\mathbf{x}(t)$. $\mathbf{B}'$ is the neural input matrix, $\mathbf{A}'$ is the neural dynamics matrix, and the transfer function $h(t)$ describes the synaptic dynamics given by the post synaptic current. Figure adapted from Eliasmith and Anderson (2003).

The principle of dynamics states that the elements of the state vector, $\mathbf{x}$ (the state variables) are the representations we find in neural populations, and the relations between these variables (defined by the system matrices) are transformations implemented by these populations. However, we need to relate the intrinsic dynamics of neural populations to this standard control theoretic characterization. To translate this into a dynamical system, the transfer function (integration in figure 3.1) needs to be a time varying description of synaptic dynamics. As mentioned before in section 3.2.1, the post synaptic current produced by the arrival of a spike is a good approximation of the synaptic dynamics, and is given by:

$$\begin{aligned} h(t) &= \frac{1}{\tau} e^{\frac{-t}{\tau}} \\ h(s) &= \frac{1}{1+s\tau} \quad \text{(frequency domain)} \end{aligned} \tag{3.9}$$

We need to characterize the neural dynamics with respect to this transfer function as shown in the 'neural' control diagram in figure 3.2. In case of a neural system, each subsystem describes a single population and hence the feedthrough and output matrices can be taken to be included in the input matrix of the subsequent population.

Taking the Laplace transform of the equations 3.8 gives:

$$s\mathbf{x}(s) = \mathbf{A}\mathbf{x}(s) + \mathbf{B}\mathbf{u}(s) \tag{3.10}$$

$$\mathbf{y}(s) = \mathbf{C}\mathbf{x}(s) + \mathbf{D}\mathbf{u}(s), \tag{3.11}$$

For a neural system, equation 3.11 will always be $\mathbf{y}(s) = \mathbf{x}(s)$ as is clear from the block diagram in figure 3.2. The neural system in the figure can be described as follows:

$$\begin{aligned} \mathbf{x}(t) &= h(t) * [\mathbf{A}'\mathbf{x}(t) + \mathbf{B}'\mathbf{u}(t)] \\ \mathbf{x}(s) &= h(s)[\mathbf{A}'\mathbf{x}(s) + \mathbf{B}'\mathbf{u}(s)] \quad \text{(frequency domain)} \end{aligned} \tag{3.12}$$

Substituting Laplace transform $h(s)$ from equation 3.9 gives:

$$\begin{aligned} \mathbf{x}(s) &= \frac{1}{1+s\tau}[\mathbf{A}'\mathbf{x}(s) + \mathbf{B}'\mathbf{u}(s)] \\ \iff s\mathbf{x}(s) &= \tau^{-1}[\mathbf{A}' - \mathbf{I}]\mathbf{x}(s) + \tau^{-1}\mathbf{B}'\mathbf{u}(s) \end{aligned} \tag{3.13}$$

where $\mathbf{I}$ is an identity matrix.

Equating right hand sides of equations 3.13 and 3.10 gives the relationship between the dynamics and input matrices for the standard control system and the neural control system as follows:

$$\begin{aligned} \mathbf{A}' &= \tau\mathbf{A} + \mathbf{I} \\ \mathbf{B}' &= \tau\mathbf{B} \end{aligned} \tag{3.14}$$

Thus, given any control system in the standard form shown in figure 3.1, we can use equation 3.14 to determine the equivalent neural system in the standard form shown in figure 3.2.

3.3 Neural Coding of probability using NEF

Recall that three different methods for neural coding of probability distributions were discussed in section 2.5. In this section, I describe how the explicit probabilistic code can be implemented using the NEF. Specifically, I adopt the approach of using basis function representation, as described by Eliasmith and Anderson (2003).

3.3.1 Representing probability distributions in neural circuits

Probability distributions are essentially functions of some parameters. In section 3.2.1, I described how vectors can be represented in neural circuits using the NEF. In this section, I describe how functions can be represented in neural circuits by considering the relationship between vector and function representation.

For any representation, we need to specify the domain of that representation. In case of vectors, the domain is the subspace of the vector space that is represented by the neurons

(e.g., the $\mathbf{x}$ vector). Function representations capture relationships between two continuous variables, and this relationship should be reflected in the neural activity. For example, if the change in pitch of a sound as a function of time is given by the function $\mathbf{x}(\upsilon)$, then the neural activity should represent the relation captured by the function $\mathbf{x}$, leading to changes in the neural activity as $\mathbf{x}(\upsilon)$ changes. To represent a function, we can define a function space analogous to a vector space. The domain of representation would then be the subspace of the function space that is represented by the neurons, i.e., $\mathbf{x}(\upsilon)$.

However if the function $\mathbf{x}(\upsilon)$ is high dimensional, then representing it in population of neurons would be very computationally expensive. For example, consider a population of neurons with N neurons and D dimensions. As explained before in section 3.2.2, when this population is connected to another D dimensional population with M neurons, instead of an $N \times M$ connection weight matrix, we have a $N \times D$ decoder matrix and a $D \times M$ encoder matrix in the NEF. In case of a high-dimensional function representation, D becomes bigger in size, and this leads to an increase in the size of encoder and decoder matrices, which explode to large sizes as D goes to infinity. Thus the neural network can become computationally expensive to simulate when representing high-dimensional functions/distributions.

Population Coding with Basis functions

In order to represent high dimensional functions in an efficient way, we can make the observation that the function spaces we are representing have a lot of redundancy. Therefore, we can find a basis space for those functions so that we need to represent only a small number of dimensions. This can be accomplished by computing the basis functions through Singular Value Decomposition. Projecting the high-dimensional function space onto these basis functions converts it to a low-dimensional space. The functions in this low-dimensional

space can then be more efficiently represented in neurons (for an example, refer to section 4.4.4: cortical2).

Formally, we can define the function domain of representation by parameterizing the set of represented functions by an n-dimensional vector of coefficients $\mathbf{k} = [k_1, k_2, \ldots, k_n]$. These define any function of interest over a fixed set of basis functions $\phi(\upsilon)$ as follows:

$$\mathbf{x}(\upsilon; \mathbf{k}) = \sum_{j=1}^{n} k_j \phi_j(\upsilon), \quad \text{for } k \sim p(\mathbf{k}). \tag{3.15}$$

Thus we define a particular probability distribution $p(\mathbf{k})$ by limiting the space spanned by the basis $\phi(\upsilon)$ to some subspace of interest depending on the application. This is also the domain over which the optimization to find the decoders in equation 3.3 is performed.

Next, we define population encoding and decoding analogous to that in equations 3.1 and 3.3 for functions:

$$a_i(\mathbf{x}(\upsilon; \mathbf{k})) = a_i(\mathbf{k}) = G_i \left[\rho_i \left\langle \mathbf{e}_i(\upsilon), \mathbf{x}(\upsilon; \mathbf{k}) \right\rangle_n + J_i^{bias} \right] \tag{3.16}$$

$$\hat{\mathbf{x}}(\upsilon; \mathbf{k}) = \sum_i a_i(\mathbf{k}) \mathbf{d}_i(\upsilon), \tag{3.17}$$

where $\mathbf{e}_i(\upsilon)$ and $\mathbf{d}_i(\upsilon)$ are the encoding and decoding functions of the neurons. We project these functions onto the same basis $\phi(\upsilon)$ used to identify the function space. For simplicity, we assume that $\phi(\upsilon)$ is an orthonormal basis – an analogous derivation for a bi-orthonormal set can be found elsewhere (Eliasmith and Martens, 2011). Hence, we get the following encoding and decoding functions:

$$\mathbf{e}_i(\upsilon) = \sum_{j=1}^{n} e_{ij}\phi_j(\upsilon) \tag{3.18}$$

$$\mathbf{d}_i(\upsilon) = \sum_{j=1}^{n} d_{ij}\phi_j(\upsilon), \tag{3.19}$$

where e_{ij} and d_{ij} identify the n coefficients that represent the encoding and decoding functions in $\phi(\upsilon)$ basis for each neuron. We now substitute the encoding function into equation 3.16:

$$\begin{aligned} a_i(\mathbf{x}(\upsilon;\mathbf{k})) &= G_i\left[\alpha_i\left(\sum_{m,n} k_n\phi_n(\upsilon)e_{im}\phi_m(\upsilon)\right) + J_i^{bias}\right] \\ &= G_i\left[\alpha_i\left(\sum_{m,n} k_n e_{im}\delta_{mn}\right) + J_i^{bias}\right] \\ &= G_i\left[\alpha_i\left(\sum_{n} k_n e_{in}\right) + J_i^{bias}\right] \\ &= G_i\left[\alpha_i\left\langle \mathbf{e}_i, \mathbf{k}\right\rangle_n + J_i^{bias}\right]. \end{aligned} \tag{3.20}$$

This way, function encoding is expressed as vector encoding identical to equation 3.5. Similarly, function decoding can also be expressed as vector decoding as follows:

$$\hat{\mathbf{k}} = \sum_i a_i(\mathbf{k})\mathbf{d}_i. \tag{3.21}$$

To summarize, I have shown that it is mathematically equivalent to talk in terms of (finite-dimensional) function spaces or (finite-dimensional) vector spaces. Since probability distributions are most generally functions, we can approximate them as high-dimensional vectors over a fixed set of basis functions using the NEF.

In section 3.2.2, we saw how transformations can be computed on vector representations in the NEF. Since distributions can be treated as vectors, the same methods can be used to compute transformations on representations of these distributions as well.

3.4 Summary

In this chapter, I presented a theoretical framework called the NEF that can be used to build neural models of cognition. The goal of this framework is to understand neural systems using the tools of engineering. The framework synthesizes different approaches in computational neuroscience, integrating ideas from neural coding, neural computation, control theory, dynamics, and probability theory.

Three principles of the NEF were presented: representation of signals by neural populations, transformations of these representations through neuronal weights called decoders, and integration of control theory and neural dynamics to understand neural systems. Additionally, the intimate relationship between the vector and function representations was shown, where function representations can be implemented as vector representations with respect to some orthonormal basis ($\phi(v)$). Since probability distributions are essentially functions, the framework provides an efficient neural code for probability distributions.

Given these characterizations, the framework can be used to understand the functional organization and operation of nervous systems from cellular level to the level of large-scale networks. I use this framework for building the neural network models presented in this thesis. In the next chapter, I use this framework to build a neural model for learning priors in a biologically plausible way, such that the learned priors can be used for Bayesian inference. The model attempts to explain how people might update their beliefs (priors) as a result of their life experience. The model is built at the algorithmic level and is

constrained by the implementation level details, e.g., synaptic time constants based on neurotransmitters, neuron properties like tuning curves, etc. In chapter 5, I show how people may use these learned beliefs to make inferences.

Chapter 4

A Spiking Neural Model for Learning Priors

4.1 Introduction

The Bayesian framework has successfully described performance in many cognitive tasks where sensory information is combined with prior assumptions. It has also been shown that the prior knowledge of statistical regularities in the environment allows humans to make accurate estimations. For example Griffiths and Tenenbaum (2006) asked people to make predictions about the duration or extent of everyday phenomena, such as human life spans or the box-office earnings of movies, and showed a close correspondence between people's implicit probabilistic models and the statistics of the world.

However, whether the priors that we use for our inferences in different situations are hard-wired or learned in response to environmental statistics is the subject of debate. A human infant is born with some biological capacities for learning that are then realized

in the environment surrounding a newborn. For example, it is generally agreed that newborns have an inherent sense of motion, space, numbers, and can distinguish animate from inanimate objects (Spelke and Kinzler 2007, Izard et al. 2009). However, the environment surrounding a newborn also supplies structured information that assists in actualizing their raw capacities. Thus, learning seems to be promoted by both the child's biology and its environment. It is thus plausible that some cortical areas represent hypotheses (sometimes learned, sometimes innate) regarding the properties of the world, which are used to interpret complex and ambiguous sensory input.

Recall from section 2.3, that the three main levels that the brain can be studied at are computational, algorithmic and implementation levels. In this chapter, I present a neural model that illustrates the interdependence of these levels of analysis. The goal of the neural model is to explore how the priors might be learned in the human brain. Specifically, I show how priors can be learned from life experience and how this learning process could be directed by existing hypotheses (assumptions at the evolution and learning levels) in the brain. I begin at the computational level specifying the nature of the problem and the goals that the brain needs to achieve, and then transition to the algorithmic and implementation levels. The resultant neural mechanisms for learning priors are not specific to a particular task, but can generalize to different inference tasks.

4.2 Cognitive Tasks

In order to demonstrate the working of the proposed neural model, I use the prediction tasks proposed by Griffiths and Tenenbaum (2006), where they evaluate how cognitive judgments compare with optimal statistical inference by asking people to predict the duration or extent of everyday phenomena. Here, I explain three of these phenomena: human life

spans, reigns of pharaohs, and waiting times on the telephone. A group of 208 people were asked to predict each of these quantities, after being presented by a question in a survey format. The responses were recorded and compared with the predictions made by a Bayesian model in each of these cases.

In the following sections, I describe each of these prediction tasks and their corresponding Bayesian models proposed by (Griffiths and Tenenbaum, 2006). Though I only use the life span inference task to demonstrate the neural models presented in this thesis, I describe two additional tasks below to show that the proposed neural model can be generalized to other scenarios as well.

4.2.1 Life spans

Subjects were asked to predict the total life spans of people, given their current age. The question was presented to them in the survey format shown below:

"Insurance agencies employ actuaries to make predictions about people's life spans – the age at which they will die based upon demographic information. If you were assessing an insurance case for an 18-year-old man, what would you predict for his life span?"

Bayesian model

If t_{total} indicates the total amount of time the person will live and t indicates his current age, the task is to estimate t_{total} from t. The Bayesian model used for this prediction task computes a probability distribution over t_{total} given t, by applying Bayes' rule:

$$p(t_{total}|t) = p(t|t_{total})p(t_{total})/p(t), \tag{4.1}$$

where:

$$p(t) = \int_0^\infty p(t|t_{total})p(t_{total})\, dt_{total}. \tag{4.2}$$

In my model, I assume the maximum age to be 120 years. Thus, when calculating $p(t)$ in practice, the integral may be computed from 0 to 120.

Prior Griffiths and Tenenbaum (2006) use publicly available real-world data to identify the true prior distribution $p(t_{total})$ over life spans. As shown in figure 4.1A, the prior over life spans is a skewed Gaussian distribution.

Likelihood The likelihood $p(t|t_{total})$ is the probability of encountering a person at age t given that their total life span is t_{total}. Griffiths and Tenenbaum (2006) assume for simplicity that we are equally likely to meet a person at any point in his or her life. As a result, this probability is uniform, and is given by:

$$\begin{aligned} p(t|t_{total}) &= 1/t_{total}, \text{ for all } t < t_{total} \text{ and} \\ &= \ 0, \text{ for } t \geq t_{total} \end{aligned} \tag{4.3}$$

Prediction function Combining the prior with the likelihood according to equation 4.1 yields a probability distribution $p(t_{total}|t)$ over all possible life spans t_{total} for a person encountered at age t. As is standard in Bayesian prediction, Griffiths and Tenenbaum (2006) use the median of this distribution—the point at which it is equally likely that the true life span is either longer or shorter—as the estimate for t_{total}. This identifies a prediction function that specifies a predicted value of t_{total} for each observed value of t.

Results Results obtained by Griffiths and Tenenbaum (2006) through this Bayesian model are shown in Figure 4.1B.

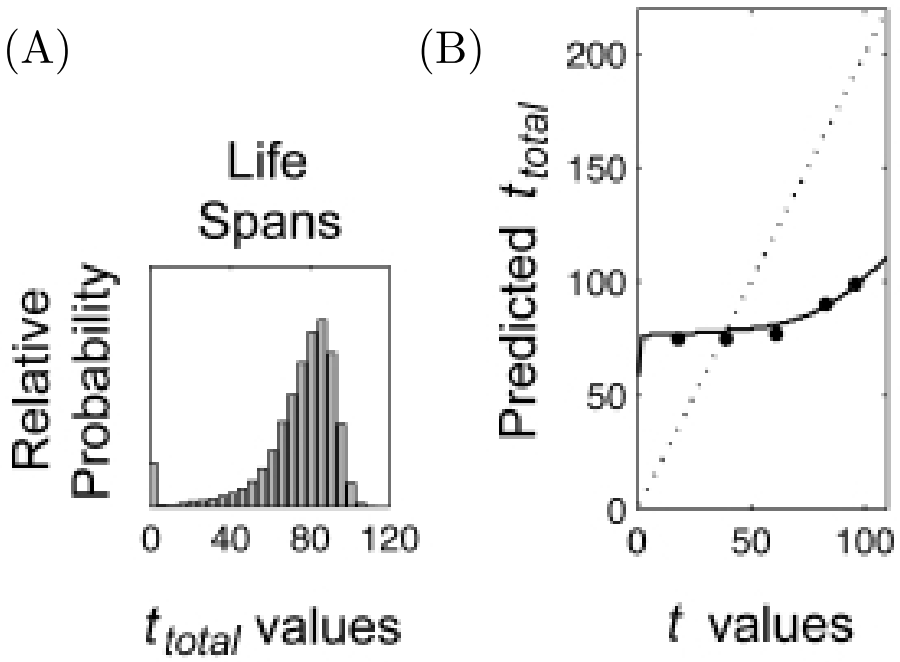


Figure 4.1: (A) Empirical distribution of the total life span t_{total}. (B) Participants' predicted values of t_{total} for a single observed sample t. Black dots show the participants' median predictions for t_{total}. Solid line shows the optimal Bayesian predictions based on the empirical prior distribution shown in A. Dotted lines show predictions based on a fixed uninformative prior. Note: the fit between the human predictions (black dots) and Bayesian predictions (solid line) looks spot on in this figure due to the compressed y-axis, but Figure 5.6 shows a zoomed version revealing that this isn't the case. Adapted from Griffiths and Tenenbaum (2006).

4.2.2 Reigns of pharaohs

Subjects were asked to predict the total reigns of pharaohs (monarchs of ancient Egypt), given the length of their reign at a particular time in history. The question was presented to them in the survey format shown below:

"If you opened a book about the history of ancient Egypt to a page listing the reigns of the pharaohs, and noticed that at 4000 BC a particular pharaoh had been ruling for 11 years, what would you predict for the total duration of his reign?"

Bayesian model

If t_{total} indicates the total amount of time a pharaoh reigned and t indicates the current length of his/her reign at a given point in history, the task is to estimate t_{total} from t. The Bayesian model used for this prediction task also computes a probability distribution over t_{total} given t, by applying Bayes' rule given by equations 4.1 and 4.2.

Prior Griffiths and Tenenbaum (2006) use the real-world data to identify the true prior distribution $p(t_{total})$ over reigns of pharaohs. This particular prior follows an Erlang distribution.

Likelihood The likelihood $p(t|t_{total})$ is the probability that the length of reign of pharaoh at a given point in history is t given that their total reign is t_{total}. Again, for simplicity, Griffiths and Tenenbaum (2006) assume that we are equally likely to encounter a pharaoh at any point in his/her reign. As a result, this probability is uniform, and given by equation 4.3.

4.2.3 Waiting times

Subjects were asked to predict the total waiting time on telephone, given the amount of time they had already been on hold. The question was presented to them in the survey format shown below:

"If you were calling a telephone box office to book tickets and had been on hold for 3 minutes, what would you predict for the total time you would be on hold?"

Bayesian model

Similar to the previous two tasks, if t_{total} indicates the total amount wait time and t indicates the amount of time one has already been on hold, the task is to estimate t_{total} from t. The Bayesian model used for this prediction tasks is also same as the previous two tasks, given by equations 4.1 and 4.2.

Prior Again, publicly available real-world data was used identify the true prior distribution $p(t_{total})$ over waiting times. It follows a Power Law distribution.

Likelihood As before, the likelihood is given by equation 4.3.

4.3 Learning Priors: Computational Level

Note that though the priors are different for the Bayesian models used for the three tasks explained above, their likelihoods follow the same expression. Griffiths and Tenenbaum (2006) show eight different tasks that have different priors, but the same likelihood. In

general, the approach to prediction used in each of the tasks explained above, is applicable to any problem in which the upper limit of a duration, extent or any other numerical quantity needs to be predicted, given a sample drawn from that interval. However, different phenomena will have different priors – one of the causes of substantial variation in the predictions. Thus, a natural question that comes to mind is: How are these priors learned? In this section, I lay the theoretical ground work for addressing this question.

I assume that priors that humans have about life spans (or pharaoh's reigns, wait times, etc.) are a result of their experiences encountering people of different ages (or different reign lengths, wait times, etc.) in their daily lives. Thus the prior will be inferred from the data that comes from life experience. I further assume that the prior is parameterized by some unknown hyperparameters α, which are to be estimated from the observed data. For instance, in the life span inference task, the observed data corresponds to the ages of m distinct people given by $\mathbf{X} = \{x_1, \ldots, x_m\}$. Here, each random variable x_i corresponds to a separate t from the previous model. Likewise, I model each element of $\mathbf{X}$ as being drawn independently from each element of $\mathbf{Z_{sampled}} = \{z_1, \ldots, z_m\}$, which correspond to the (unknown or hidden) life spans of these same m people. Each random variable z_i is drawn (or "sampled") from the unknown prior, and corresponds to a separate t_{total} from the previous model. See figure 4.2 and section 4.4.2 for a detailed explanation. I now describe three standard methods for determining a prior from observed data $\mathbf{X}$, by obtaining an estimate $\hat{\alpha}$ of the hyperparameters.

4.3.1 Marginal likelihood

If we do not know the actual prior, then the optimal solution can be found by trying them all. That is, I directly find the hyperparameters $\hat{\alpha}$ that maximize the marginal likelihood

of the observed data, $L(\alpha; \mathbf{X})$ (or equivalently the log-likelihood for numerical stability):

$$\begin{aligned} L(\alpha; \mathbf{X}) &= p(\mathbf{X}|\alpha) \\ &= \prod_{i=1}^{m} p(x_i|\alpha) \\ &= \prod_{i=1}^{m} \sum_{z_i} p(x_i, z_i|\alpha) \\ \implies \hat{\alpha} &= \text{argmax}_\alpha L(\alpha; \mathbf{X}) \\ &= \text{argmax}_\alpha \log L(\alpha; \mathbf{X}) \\ &= \text{argmax}_\alpha \sum_{i=1}^{m} \log \sum_{z_i} p(x_i, z_i|\alpha). \end{aligned} \tag{4.4}$$

Here, each observed sample $x_i \in \mathbf{X}$ and $z_i \in \mathbf{Z}$ where $\mathbf{Z}$ contains all possible values of total life spans, which I assume to be from 1 to 120 i.e., $\mathbf{Z} = \{1, 2, \ldots, 120\}$. I also assume that $\mathbf{X} \subset \mathbf{W}$ where $\mathbf{W}$ is the set of whole numbers.

In general, however, the procedure described above is intractable, since it requires that we iterate over all combinations of α and $\mathbf{Z}$. This motivates near-optimal iterative procedures such as the widely-used expectation maximization algorithm (Dempster et al., 1977).

4.3.2 Expectation Maximization

Below I work out the details of the Expectation Maximization (EM) procedure for the case where the hyperparameters are $\alpha = (\mu, \sigma^2)$, i.e., the prior is assumed to be normally distributed with unknown moments. I begin by simplifying the expectation function using independence and other known facts about the model:

$$
\begin{aligned}
Q(\alpha|\alpha^{(t)}) &= E_{\mathbf{Z}|\mathbf{X},\alpha^{(t)}}\left[\log L(\alpha;\mathbf{X},\mathbf{Z})\right] \\
&= \sum_{i=1}^{m} E_{z_i|x_i,\alpha^{(t)}}\left[\log L(\alpha;x_i,z_i)\right] \\
&= \sum_{i=1}^{m}\sum_{z_i} p(z_i|x_i,\alpha^{(t)}) \log p(x_i,z_i|\alpha) \\
&= \sum_{i=1}^{m}\sum_{z_i} T(x_i,z_i) \log\left(p(x_i|z_i)p(z_i|\alpha)\right) \\
&= \sum_{i=1}^{m}\sum_{z_i} T(x_i,z_i) \log\left(p(z_i|\alpha)/z_i\right),
\end{aligned}
\tag{4.5}
$$

where I have defined:

- $T(x_i, z_i) := p(z_i|x_i, \alpha^{(t)})$ to be some fixed function with respect to $\alpha^{(t)}$.
- $p(x_i, z_i|\alpha) = p(x_i|z_i)p(z_i|\alpha)$, with $\alpha^{(t)}$ included, since $p(x_i|z_i, \alpha) = p(x_i|z_i)$ by assumption.
- Substituting $p(x_i|z_i) = 1/z_i$ for all z_i is acceptable since $T(x_i, z_i) = 0$ for $z_i \leq x_i$ by definition.

Next, I simplify the log expression using the assumed model of the prior, with unknown $\alpha = (\mu, \sigma)$:

$$
\begin{aligned}
\log\left(p(z_i|\alpha)/z_i\right) &= \log\left(\frac{1}{\sqrt{2\sigma^2\pi}} e^{-\frac{(z_i-\mu)^2}{2\sigma^2}}\right) - \log z_i \\
&= -\frac{1}{2}\left((z_i-\mu)^2/\sigma^2 + \log\sigma^2 + \log(2\pi) + 2\log z_i\right),
\end{aligned}
\tag{4.6}
$$

and then differentiate this with respect to μ:

$$\frac{\partial \log\left(p(z_i|\alpha)/z_i\right)}{\partial \mu} = (z_i - \mu)\sigma^2, \tag{4.7}$$

and with respect to σ^2:

$$\begin{aligned}\frac{\partial \log\left(p(z_i|\alpha)/z_i\right)}{\partial \sigma^2} &= \frac{1}{2}(z_i - \mu)^2/\sigma^4 - \frac{1}{2}/\sigma^2 \\ &= \frac{1}{2}\left((z_i - \mu)^2 - \sigma^2\right)/\sigma^4.\end{aligned} \tag{4.8}$$

By linearity of differentiation, the derivatives of $Q(\cdot)$ are zero when:

$$\begin{aligned}\frac{\partial Q(\alpha|\alpha^{(t)})}{d\mu} &= \sum_{i=1}^{m}\sum_{z_i} T(x_i, z_i)(z_i - \mu)\sigma^2 = 0 \\ \iff \quad \mu &= \frac{\sum_{i=1}^{m}\sum_{z_i} z_i T(x_i, z_i)}{\sum_{i=1}^{m}\sum_{z_i} T(x_i, z_i)},\end{aligned} \tag{4.9}$$

and similarly:

$$\begin{aligned}\frac{\partial Q(\alpha|\alpha^{(t)})}{d\sigma^2} &= \sum_{i=1}^{m}\sum_{z_i} T(x_i, z_i)\frac{1}{2}\left((z_i - \mu)^2 - \sigma^2\right)/\sigma^4 = 0 \\ \iff \quad \sigma^2 &= \frac{\sum_{i=1}^{m}\sum_{z_i}(z_i - \mu)^2 T(x_i, z_i)}{\sum_{i=1}^{m}\sum_{z_i} T(x_i, z_i)}.\end{aligned} \tag{4.10}$$

Finally, the generalized Bayes' rule gives:

$$T(x_i, z_i) = p(z_i|x_i, \alpha^{(t)}) = \frac{p(z_i, x_i|\alpha^{(t)})}{\sum_{z_i} p(z_i, x_i|\alpha^{(t)})},$$

which may be computed via Eq. 4.1. Also since $T(\cdot)$ is a probability density function over z_i, note that:

$$\sum_{i=1}^{m}\sum_{z_i} T(x_i, z_i) = \sum_{i=1}^{m} 1 = m.$$

Therefore, each EM iteration must make the update $\alpha^{(t+1)} = (\mu^{(t+1)}, \sigma^{(t+1)})$, where:

$$\begin{aligned}\mu^{(t+1)} &= \frac{1}{m}\sum_{i=1}^{m}\frac{\sum_{z_i} z_i p(z_i, x_i|\alpha^{(t)})}{\sum_{z_i} p(z_i, x_i|\alpha^{(t)})} \\ \sigma^{(t+1)} &= \sqrt{\frac{1}{m}\sum_{i=1}^{m}\frac{\sum_{z_i}(z_i - \mu^{(t+1)})^2 p(z_i, x_i|\alpha^{(t)})}{\sum_{z_i} p(z_i, x_i|\alpha^{(t)})}}.\end{aligned} \tag{4.11}$$

This converges to some locally optimal estimate of the hyperparameters. For initial $\alpha^{(0)}$ chosen sufficiently close to the global optimum $\hat{\alpha}$ given by equation 4.4, this converges to the optimum.

This provides a tractable procedure for updating the prior. In particular, I begin with some initial guess at the hyperparameters, and then update them iteratively to better explain the observed data. In practice only a few iterations are required (results not shown). Once I have an estimate of the hyperparameters $\hat{\alpha}$, the prior $p(t_{total}|\hat{\alpha})$ is then known.

4.3.3 Online Expectation Maximization

Though EM provides a tractable procedure for updating the prior, each iteration in EM requires all observed ages of m distinct people (equation 4.5). In other words, all observations $\mathbf{X} = \{x_1, \ldots, x_m\}$ are required to even begin learning. This seems counterintuitive for human cognition since humans do not wait for all or m number of observations before they start learning. On the contrary, I presume humans usually learn from their experiences as they encounter them.

This further motivates online iterative procedures such as the incremental and stepwise expectation maximization algorithms (Liang and Klein, 2009). These algorithms are

stochastic versions of the EM algorithm, and allow updating of parameters with each incoming observation. One weakness of incremental EM is that its memory requirements grow linearly with the number of observations. Since humans have limited memory, incremental EM does not satisfy the memory constraints imposed by the brain. Memory requirements of Stepwise EM, on the other hand, remain constant with an increasing number of observations, making it a suitable choice. Note that this is an example where the implementation level is providing constraints for the computational level description.

Formulation for Stepwise EM is as follows:

$$\begin{aligned}
&\lambda \leftarrow \text{initialization; } l = 0 \\
&\text{foreach observation i = 1,...,m :} \\
&\quad s_i' = \sum_{z_i} T(x_i, z_i) \log\left(p(z_i|\alpha)/z_i\right) \\
&\quad m_l = (l+2)^{-\beta} \\
&\quad \lambda = (1 - m_l)\lambda + m_l s_i' \\
&\quad \alpha^{(t)} = \text{argmax}(\lambda), \quad l \leftarrow l + 1
\end{aligned} \tag{4.12}$$

According to the generalized Bayes rule:

$$\begin{aligned}
T(x_i, z_i) = p(z_i|x_i, \alpha^{(t)}) = \frac{p(z_i, x_i|\alpha^{(t)})}{\sum_{z_i} p(z_i, x_i|\alpha^{(t)})} \\
\text{where:} \\
p(z_i, x_i|\alpha^{(t)}) = p(x_i|z_i)p(z_i|\alpha^{(t)})
\end{aligned} \tag{4.13}$$

Here s_i' is the sufficient statistics for the i-th observation and λ is the weighted sum of statistics corresponding to the current observation and the previous observations. Instead

of summing these over m observations as in the regular EM algorithm (equation 4.5), here I interpolate between λ and s'_i based on a stepsize m_l, where l is the number of updates made to λ so far. Thus Stepwise EM leads to a stochastic process where I approximate the update in each iteration with a single sensory observation. However, since a single sample is a bad approximation, I interpolate between the current observation (s'_i) and the set of statistics corresponding to the previous observations (current λ). According to the results from stochastic approximation literature, $\sum_{l=0}^{\inf} m_l = \inf$ and $\sum_{l=0}^{\inf} m_l^2 < \inf$ are sufficient conditions to guarantee convergence to a local optimum. This implies that for $m_l = (l+2)^{-\beta}$, any value of β such that $0.5 < \beta <= 1$ is valid. In general, smaller β would lead to larger updates and quicker decay of the old sufficient statistics λ. Thus, small β can lead to fast progress but can also generate instability (Liang and Klein, 2009). In my model, I use $\beta = 0.65$.

In the next section, I describe neural mechanisms for implementing the iterative process of the stepwise EM algorithm. Now that I have a computational level description of the problem at hand, I can think about the neural representations (algorithmic level), and use the knowledge at the implementation level (e.g., structure of neurons, synapses, plausible networks) to constrain the possible computational operations and algorithms that can be employed at the algorithmic level.

4.4 Neural Model

Figure 4.5 shows a schematic diagram of the neural model built for learning priors. In the following sections, I first describe the prior space - a set of distributions assumed at the evolution and learning levels; second, I describe the training data used to train the model; and finally, I describe the neural model implemented at the algorithmic level constrained

by the implementation level details.

4.4.1 Prior space

From the evolution level (section 2.3), we learned that humans are able to learn very effectively from a few examples, partly due to the huge bank of prior knowledge. Deriving from this, I assume that rather than learning completely from scratch, given either evolutionary history, or personal history, or most likely a combination of both, our brain has a characterization of the set of distributions most likely to capture real-world data. Thus, I assume a set of prior distributions defined as a prior space of size M. These distributions belong to three different families – Gaussian, Erlang and Power Law and the task is to estimate which one of them is the prior corresponding to the total life spans. These three families of distributions were chosen based on the psychological evidence presented by Griffiths and Tenenbaum (2006), where they suggest that the prior beliefs informing people's predictions on a variety of tasks follow these distributions. For example, in section 4.2, we saw that the priors people use for predicting life spans, Pharaoh's reigns and waiting times on telephone fall into these families. Some other tasks with priors from these families include predicting poem lengths, and movie box office earnings.

Though I demonstrate the model using the life span inference task, the same model can also be used to learn the priors for any of the tasks that have priors in these families, e.g., predicting the waiting times on telephone. In short I suggest a general approach that is applicable to any problem that requires estimating the upper limit of a numerical quantity given a sample drawn from that interval, and that has a Gaussian, Erlang or Power Law prior. However, this approach can be further generalized as discussed in sections 6.3 and 6.4. For instance, the distributions in the prior space can be used as basis functions to

learn priors which are their linear combinations. To make predictions for unfamiliar events (for which they do not have sufficient direct experience), people might also use an adaptive strategy where they may be able to identify the appropriate form of the distribution by making an analogy to a more familiar phenomena in the same broadly defined class.

To construct the prior space, I randomly sample the hyperparameters of the three families of distributions from a range chosen based on the psychological evidence presented by Griffiths and Tenenbaum (2006). The probability density functions and the corresponding parameter ranges are listed below.

1. A skewed form of the Gaussian distribution is used having three hyperparameters with ranges: mean (μ) (89, 109), scale (σ) - a measure of std-deviation - (5, 20) and shape (γ) - a measure of skewness - (-8, -4). The probability density function is as follows:

   Std. Gaussian probability density function (PDF) is given by:

   $$\phi(z;\mu,\sigma) = \frac{\exp^{-\frac{(z-\mu)^2}{2\sigma^2}}}{\sqrt{2\pi\sigma^2}}$$

   Std. Gaussian cumulative distribution function (CDF) is given by:

   $$\Phi(z;\mu,\sigma) = \int_{-\inf}^{z} \phi(z)dz = \frac{1}{2}[1 + erf(\frac{z-\mu}{\sigma\sqrt{2}})]$$

   where $erf(z)$ is the error function.

   Then the probability density function of the skewed Gaussian is given by:

   $$f(z;\mu,\sigma,\gamma) = 2\phi(z)\Phi(\gamma z) \quad \text{where } \mu \in (89,109),\ \sigma \in (5,20),\ \gamma \in (-8,-4). \quad (4.14)$$

2. The Power law distribution used has a single hyperparameter: shape(γ) ranging (1, 5). The PDF of the power law is as follows:

$$f(z;\gamma) = cz^{-\gamma} \quad \text{where } \gamma \in (1,5) \text{ and } c \text{ is a constant.} \tag{4.15}$$

3. The Erlang distribution used has two hyperparameters with ranges: scale (σ) (25, 35) and shape (γ) fixed at 2.0 following Griffiths and Tenenbaum (2006) and Shepard (1987). The PDF of the Erlang distribution is as follows:

$$f(z;\gamma,\sigma) = \frac{z^{\gamma-1}\exp^{-z/\sigma}}{\sigma^{\gamma}(\gamma-1)!} \quad \text{where } \gamma = 2 \text{ and } \sigma \in (25,35). \tag{4.16}$$

I assume an equal number of distributions from each family to construct the prior space. In other words, for a prior space of size M, the number of distributions belonging to each family is $M/3$. In section 4.4.5, I present model results for two different prior spaces of sizes $M = 27$ and $M = 108$.

4.4.2 Training data

Recall that one of the questions asked at the learning level (section 2.3) is: How do humans learn from experiences of the natural world? To show how humans might learn from their experiences, I assume that the neural model receives one observed age x_i as input at a time (as a human would on encountering people of different ages). The model uses this observation to compute an initial estimate of the prior over the total life spans of people, and keeps iterating (according to the Stepwise EM algorithm explained in section

4.3.3) to improve the prior estimate until the next observation is received as input. As it receives more observations, it interpolates between the current observation and statistics from previous observations (stored in memory) to estimate the optimal prior.

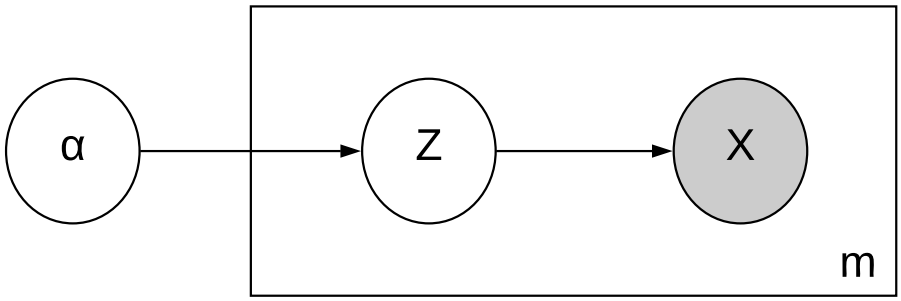


Figure 4.2: Hierarchical Bayesian Network showing dependencies between variables using the plate notation. These dependencies are used to generate the training data for the neural model. The variables in the shaded circles are observable to the neural model for training, while the empty circles represent latent variables.

The observations x_i used to train the model are generated using a two-stage hierarchical Bayesian network shown in figure 4.2. The Bayesian network is depicted using the plate notation, where the rectangle or plate is used to group the variables in the model that repeat together, with number of repetitions of the subgraph in the plate written on the bottom right corner. Shaded circles indicate observable variables, i.e., ones that would be visible to the neural model for training, and the non-shaded circles indicate latent variables, i.e., variables that are not directly observed but are inferred by the neural model. The directed edges indicate dependencies between variables, for instance, in this case each element of $\mathbf{X}$ is conditionally dependent on each element of $\mathbf{Z}_{\mathbf{sampled}}$, which in turn is sampled from the true prior conditioned on the set of hyperparameters α. This is detailed in the next

paragraph.

Recall from section 4.3, that the prior is parameterized by a set of hyperparameters α and the goal is to infer them from the observed ages of m distinct people given by $\mathbf{X} = \{x_1, \ldots, x_m\}$. I generate the observed data in the forward direction ($\alpha \longrightarrow Z \longrightarrow X$). Each element of $\mathbf{Z_{sampled}} = \{z_1, \ldots, z_m\}$ corresponds to the unknown life spans of the same m people, and is a random sample drawn from the true prior distribution (conditioned on α). Each element of $\mathbf{X}$ is drawn conditionally from each element of $\mathbf{Z_{sampled}}$. I model $x_i|z_i \sim \mathcal{U}[0, z_i)$, meaning that we are equally likely to see a person at any time throughout their life (based on the likelihood assumption stated in equation 4.3).

4.4.3 Neural Network Design

The goal of this section is to map the computational description for learning priors described in section 4.3.3 to an algorithmic level description that is constrained by the implementation level details of the brain.

Looking at the computational level description, the following come to mind:

- First, I need to define neural representations (encoding and decoding scheme) for probability distributions. This was already defined in section 3.3.
- Second, I need a way to store the weighted sum λ such that it can be maintained across different observations. Information is usually stored in the brain in the form of memories. Therefore, one possibility is to use a memory network to store λ. Such a network can easily be built using principle 3 of the NEF (explained in section 3.2.3).
- Third, I need a way to implement the $argmax()$ function in the neural network. Winner-take-all (WTA) networks are commonly used neural mechanisms for selecting

the largest value among a number of competing values. Several proposals for such networks can be found in the cognitive science literature, and one or more of them may be suitable for our purposes.

Now that I have identified the kinds of circuits or sub-networks that I would need in my neural network, I can choose the specific version of these networks that are most suitable for my model.

Memory Network

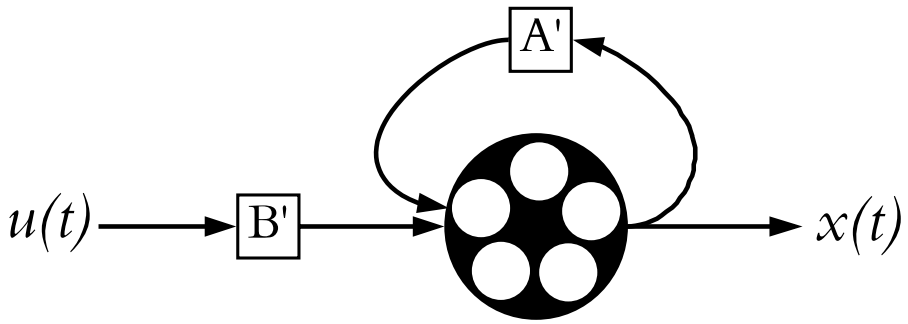


Figure 4.3: Schematic diagram of a neural integrator implemented using the NEF. The circle represents a population of neurons and the square boxes represent the constant weight matrices applied to the signals. Weight matrices for an integrator are given by: $\mathbf{A}' = 1$ and $\mathbf{B}' = \tau_{PSC}$.

The most basic memory circuit that can be implemented using a neural network is an integrator. An integrator sums a signal over time by continuously adding a given input signal to its internal state. When the input signal is removed, the integrator maintains

the internal state it had reached, thereby functioning like a memory. The dynamics of an integrator in standard control theoretic form can be expressed as follows:

$$\dot{\mathbf{x}}(t) = \mathbf{A}\mathbf{x}(t) + \mathbf{B}\mathbf{u}(t) \tag{4.17}$$

where $u(t)$ is the input vector, $\mathbf{x}(t)$ is the output vector that represents the current state of the integrator, and $\mathbf{A}$ and $\mathbf{B}$ are the dynamics matrix and the input matrix respectively. For an integrator, $\mathbf{A} = 0$ and $\mathbf{B} = 1$.

Recall from section 3.2.3 that the NEF can be used to convert any differential equation into its biologically plausible spiking neural network approximation, using the principle of dynamics. I now use equation 3.14 to derive the dynamics matrix **A'** and the input matrix **B'** for the corresponding neural control system:

$$\begin{aligned} \mathbf{A}' &= \tau\mathbf{A} + \mathbf{I} = \tau 0 + \mathbf{I} = 1 \\ \mathbf{B}' &= \tau\mathbf{B} = \tau 1 = \tau \end{aligned} \tag{4.18}$$

where τ is the time constant of the post synaptic current (PSC) of the neural population representing $\mathbf{x}(t)$. The network diagram of the resulting neural integrator is shown in figure 4.3.

Though the neural integrator I have built is capable of acting as a memory unit, the value $\mathbf{x}(t)$ that it holds is dependent on the time for which the input is presented to the circuit. In other words, presenting a square pulse of height 1 as input for half a second leads to a stored value of 0.5 in the integrator. However, presenting the same pulse for 1 second leads to a stored value of 1 in the integrator. What I want instead, is a circuit that can hold its input (target value) regardless of how long its presented for. Choo (2010) has proposed a gated difference integrator network to accomplish this.

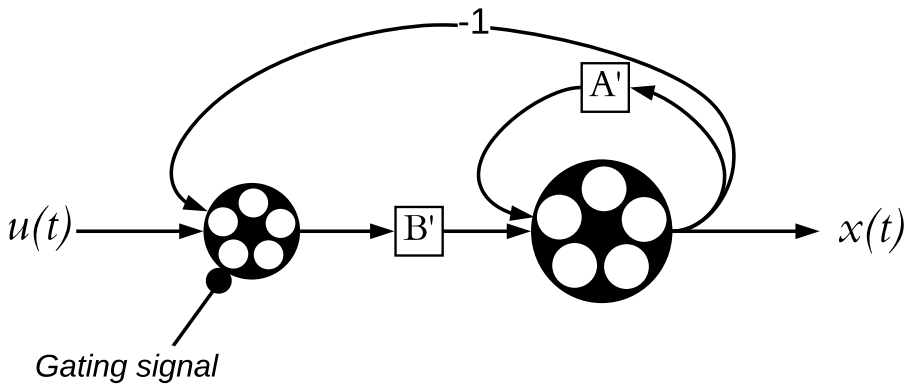


Figure 4.4: A schematic diagram of the gated difference integrator implemented using the NEF. The circles represent populations of neurons and the square boxes represent the constant weight matrices $\mathbf{A}' = 1$ and $\mathbf{B}' = \tau_{PSC}$ applied to the signals. The smaller circle is the gating population and the bigger circle is the integrator population. Full circle(—•) indicates an inhibitory connection which can be used for inhibiting the gating population to enable the integrator to hold the previous value when the current input $\mathbf{u}(t)$ is zero.

A gated difference integrator is shown in figure 4.4. It consists of two neural populations, a primary integrator population, and a gating population. The basic principle is to provide an additional negative feedback to the integrator. The input to the integrator population is provided through a gating population which also receives the negative of the currently stored value. Thus the total input to the integrator population is the difference of the current value and the target value $\mathbf{u}(t)$. Due to this, the integrator continues to receive an input signal only until its state matches the target value. Once, the state of the integrator matches the target value, its total input becomes zero (since the negative feedback fully cancels the target signal $\mathbf{u}(t)$), and the integrator holds its value. The gating population can also be inhibited to disable any input to the integrator. Without any input, a recurrent

connection with a long synaptic time constant ($\tau = 0.1s$) ensures that the currently stored value doesn't change much over time (except for a small drift due to noise).

To summarize, a gated difference integrator is capable of holding a target value regardless of the amount of time it is presented for, and is therefore a suitable network for storing the weighted sum λ in our neural model.

Winner-Take-All Network

A winner-take-all (WTA) mechanism determines the magnitude and identity of its largest input. Such mechanisms have been proposed for various brain functions and have been found to be consistent with a large body of psychological and neurobiological literature (Smith and Ratcliff 2004, Bogacz et al. 2006). WTA networks are very commonly used as components of cognitive models specially in models of action selection, attention (Lee et al. 1999, Itti et al. 1998, Koch and Ullman 1987) and decision making (Usher and McClelland 2001, O'Reilly 1998, Furman and Wang 2008).

For the purpose of my neural model, I use the winner-take-all mechanism called the spiking independent accumulator (IA) model proposed by Gosmann et al. (2017). The network diagram of the IA model is shown in Figure 4.5 (labelled "WTA Network"). It is a two-layer spiking neural network built using the principles of the NEF. Given an n dimensional input vector corresponding to the non-negative utilities of n different choices, the desired output of the IA network is positive for the dimension with the highest utility ("winner") and zero for all others. To accomplish this, the first layer of the network consists of n separate integrator populations (same as shown in figure 4.3) for accumulating each input, and the second layer consists of n non-recurrent thresholding populations that receive one-to-one input from the neural populations in the first layer. A thresholding neural

population contains neurons which respond only to inputs above a certain threshold.

Note that there are no direct interactions among the accumulators in the first layer of the network (i.e., they are 'indepenent'). This is in contrast to some other winner-take-all models which have direct competition between states, e.g, the leaky competing accumulator model (Usher and McClelland, 2001). The competition in the IA model is enabled through the second thresholding layer. Each neural population in the second layer projects back to mutually inhibit all other neural populations in the first layer except the one it receives input from, which it excites. Thus the largest input accumulates the fastest, and once it reaches the threshold, it self excites and inhibits the accumulators corresponding to all other inputs.

4.4.4 Model Implementation

I now use the components explained above to build the neural model for learning priors, and demonstrate its function using the the life span inference task described in section 4.2.1. Figure 4.5 shows the final network diagram of the neural model. Circles indicate neural populations that represent information in spiking LIF neurons using principle 1 of the NEF (section 3.2.1). The arrows connecting the neural populations are neural connections that implement transformations and compute functions using the principle 2 of the NEF (section 3.2.2). The dynamics of the recurrently connected neural populations are specified using the principle 3 of the NEF (sections 3.2.3 and 4.4.3). The number of neurons and dimensions represented by each neural population or sub-network are labelled as N and D respectively. M indicates the size of the prior space used (explained in section 4.4.1). The model was built using Nengo, a neural simulator based on the NEF (Bekolay et al., 2014). In table 4.1, I briefly describe where each computation of the Stepwise EM algorithm given

by equation 4.12 is implemented in the model, followed by details of neural implementation under respective model components.

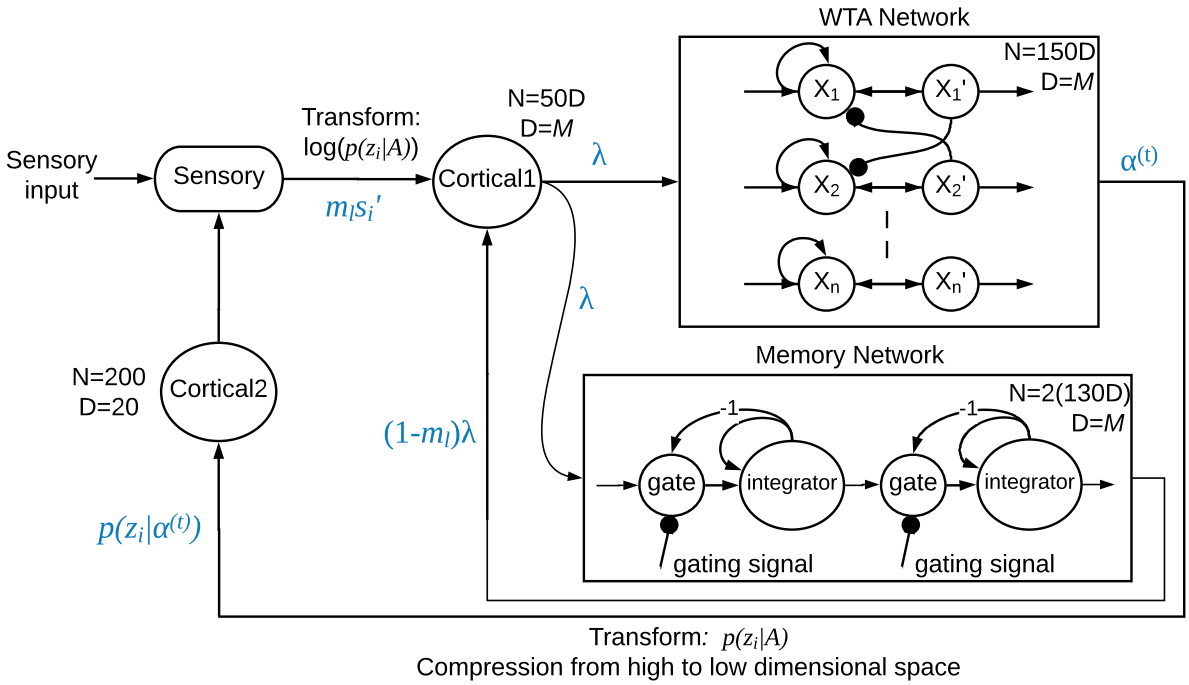


Figure 4.5: A schematic diagram of the neural model for learning priors. The circles represent neural populations/networks. The rectangles group the neural populations into subnetworks. The rounded box represents a node (non-neural). The flow of information is denoted with arrows (→). Transformations are labelled along these connections. Inhibitory connections are indicated with full circles(—•). The blue labels indicate the quantity encoded by the spike trains propagating through the connections. The number of neurons and dimensions represented by each population/network are labelled as N and D respectively (M is the prior space size).

Table 4.1: Mapping the Stepwise EM algorithm to the neural model.

Stepwise EM	**Neural Model**
$s'_i = \sum_{z_i} T(x_i, z_i) \log\left(p(z_i\|\alpha)\right)$	Computed as a dot product between $T(x_i, z_i)$ and $\log\left(p(z_i\|\mathbf{A})\right)$, where $\mathbf{A} = \{\alpha_1, \alpha_2 \ldots \alpha_M\}$. Each α_i corresponds to a different prior distribution in the prior space. $T(x_i, z_i)$ is computed in the `sensory` node. $\log\left(p(z_i\|\mathbf{A})\right)$ is the transform on the connection between the `sensory` node and `cortical1` neural population. s'_i is M dimensional, containing utilities corresponding to each of the priors in the prior space.
$m_l = (l+2)^{-\beta}$	Computed in the `sensory` node. β is a constant parameter set to 0.65 (explained in section 4.3.3).
$\lambda = (1 - m_l)\lambda + m_l s'_i$	$m_l s'_i$ is computed in the `sensory` node. $(1 - m_l)\lambda$ is computed in a node (not shown in figure 4.5) that lies between the the memory network and `cortical1`. This node receives the output from the memory network (λ) as input. The addition happens in the `cortical1` neural population. λ is M dimensional, and contains utilities corresponding to each of the priors in the prior space.
$\alpha^{(t)} = \text{argmax}(\lambda)$	Computed by the WTA network.
$l \leftarrow l + 1$	Updated in the `sensory` node at the beginning of each iteration. The length of each iteration was set to 1 second. The model keeps iterating (to improve it's estimate) with the current utilities for 1 second, after which a new observation is received as input and used to update the utilities.

Training data: I assume 120 possible values of total life spans i.e., $\mathbf{Z} = \{1, 2, \ldots 120\}$. This implies that $\mathbf{Z_{sampled}} = \{z_1, z_2, \ldots z_m\} \subset \mathbf{Z}$ and each of its elements is drawn randomly from $\mathbf{Z}$, based on their probability from the true prior distribution over $\mathbf{Z}$ (or t_{total}). This 120 dimensional true prior distribution over the life spans is shown in figure 4.1. Each element of $\mathbf{X} = \{x_1, x_2, \ldots x_m\}$ is drawn conditionally from each element of $\mathbf{Z}$ as explained in section 4.4.2. The model successively receives x_i as input (one sample at a time). In the rest of this section, I use z_i to refer to the elements of $\mathbf{Z}$ (not the elements of $\mathbf{Z_{sampled}}$). I also assume that $\mathbf{X} \subset \mathbf{W}$.

Prior space: Prior space is constructed as explained in section 4.4.1. It consists of M priors conditioned on their respective hyperparameters α. Each prior is 120 dimensional and gives $p(z_i|\alpha)$, i.e, a possible prior probability distribution over the total lifespans (see figure 4.6 for examples of such distributions). The prior space is represented as a $120 \times M$ dimensional matrix and we will refer to it as $p(z_i|\mathbf{A})$, where $\mathbf{A} = \{\alpha_1, \alpha_2 \ldots \alpha_M\}$. Each α_i corresponds to a different prior distribution in the prior space.

Sensory node

As shown in figure 4.5, the `sensory` node receives the input x_i and computes $T(x_i, z_i) * m_l$ using equation 4.13. Computing $T(x_i, z_i)$ requires computing the posterior distribution $p(z_i|x_i, \alpha^{(t)})$ as a product of the likelihood $p(x_i|z_i)$ and the prior $p(z_i|\alpha^{(t)})$. The denominator $\sum_{z_i} p(z_i, x_i|\alpha^{(t)})$ is just a normalizing factor and can therefore be ignored.

For the first iteration, the model uses a random initial guess for the prior (or $\alpha^{(t)}$) to compute $p(z_i|\alpha^{(t)})$. In simple words, it randomly chooses an initial prior from the prior space.

The likelihood is computed using the following equation (explained before in section 4.2.1). It is a 120 dimensional distribution that gives the likelihood of each $z_i \in \mathbf{Z}$ given the current observation x_i.

$$p(x_i|z_i) = 1/z_i, \text{ for all } x_i < z_i \text{ (and 0 for } x_i \geq z_i) \tag{4.19}$$

Once I have the prior and the likelihood, $T(x_i, z_i)$ is simply a 120 dimensional vector given by the element-wise product of the two. Note that the 'node' object in Nengo does not contain neurons and performs computations mathematically. However, this computation has been abstracted here only for simplification purposes. It is possible to compute this product using spiking neurons in a neurally plausible way, as will be shown later in chapter 5. The resulting product is then multiplied by the scalar m_l to get $T(x_i, z_i) * m_l$.

Cortical1

The connection from the `sensory` node to the `cortical1` neural population computes the transform $\log(p(z_i|\mathbf{A}))$, i.e., the $120 \times M$ prior space matrix. This matrix is used as the weight matrix on the connection and the transformation $\langle (T(x_i, z_i) * m_l)_{120}, (p(z_i|\mathbf{A}))_{120*M} \rangle_M$ is computed using principle 2 of the NEF. The notation $\langle \cdot, \cdot \rangle_M$ indicates a dot product or a matrix vector product and the subscripts denote the dimensionality. This input to `cortical1` is thus an M dimensional vector, where each dimension represents the utility of the corresponding prior in the prior space based on the current observation ($m_l s'_i$ from equation 4.12).

To this point I have computed $m_l s'_i$ where s'_i is given by (rewriting from equation 4.12):

$$
\begin{aligned}
s'_i &= \sum_{z_i} T(x_i, z_i) \log\left(p(z_i|A)/z_i\right) \\
&= \sum_{z_i} T(x_i, z_i) \log\left(p(z_i|A)\right) - \sum_{z_i} T(x_i, z_i) \log\left(z_i\right)
\end{aligned}
\tag{4.20}
$$

Notice that in my computation of s'_i I have not accounted for the subtraction term in equation 4.20. This is because this term leads to the dot product $\langle (T(x_i, z_i))_{120}, (log(z_i))_{120} \rangle_{120}$, which gives a scalar value. Since the same scalar value gets subtracted from each of the terms in the M dimensional vector (given by the first term in equation 4.20), subtracting this scalar doesn't make any difference to their values relative to each other. Recall that I only care about the relative values of the elements in the M dimensional vector (each of which correspond to a different prior in the prior space), since my goal is to compare them and pick the most optimal prior.

The other input to `cortical1` is feedback from the memory network, which provides utilities based on the statistics from the previous observations ($(1 - m_l)\lambda$ from equation 4.12). For the first observation x_1, the feedback from memory is zero since nothing has been stored there yet. To summarize, `cortical1` is an M dimensional neural population that represents λ and also sends it to the memory network which stores it for future use. It represents the M dimensional vector λ using 50 LIF neurons for each dimension.

Memory Network

The Memory network consists of two gated difference integrator units as shown in Figure 4.5. Each unit contains two neural populations per dimension, a primary population `integrator` that has 100 neurons and a `gate` population that has 30 neurons. Each of the units can act as a memory to store a target value as explained in section 4.4.3. The

two units are gated out of phase such that when the first unit is open (receives input from `cortical1`), the second unit is closed and sends its stored value to `cortical1` in order to add the summary statistics from previous observations to the current utilities. As a result, the first unit gets updated summary statistics as input. Next, the first unit closes and the second opens to shift the summary statistics to the second unit for the next iteration. Note that the feedback from the memory network λ is multiplied by the factor $1 - m_l$ through a node (not shown in the schematic), before it is sent to the `cortical1` population.

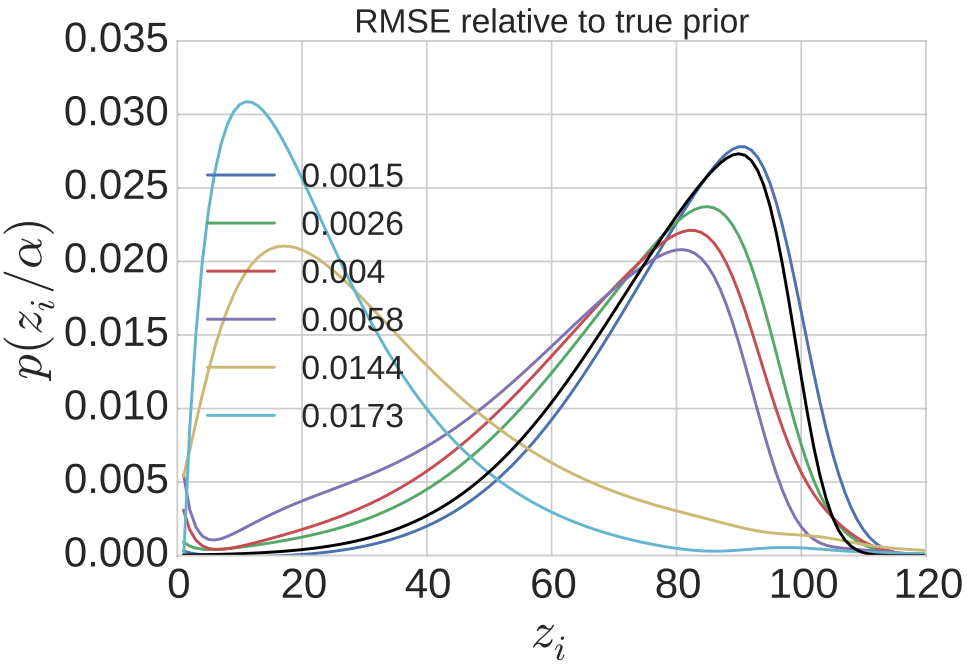


Figure 4.6: Root mean squared error (RMSE) of distributions relative to the true prior (shown in black). Power Law distributions (not shown) have RMSE ranging from 0.03 to 0.05 relative to this prior. Note that the model picks these distributions during training, i.e., the optimal distribution evolves over time.

WTA Network

The other output connection from `cortical1` provides the M dimensional input to the independent accumulator WTA network (explained in section 4.4.3) that is used to implement the $argmax(\lambda)$ operation in equation 4.12. The IA network is a two-layer spiking neural network whose first layer consists of M separate integrator populations with 100 neurons each, and the second layer consists of M non-recurrent thresholding populations with 50 neurons each. Recall from section 4.4.3 that the output of the IA network is positive for the dimension with the highest utility (winner) and zero for all others. The output connection from the WTA network uses the principle 2 of the NEF to compute the transform $p(z_i|\mathbf{A})$. In other words the $120 \times M$ dimensional prior space matrix, $p(z_i|\mathbf{A})$ is used as the weight matrix on this connection. The dot product of the WTA output with this matrix gives the current optimal prior distribution corresponding to $\alpha^{(t)}$ i.e., $p(z_i|\alpha^{(t)})$ which is the input to the `cortical2` population.

Cortical2

The 120 dimensional prior $p(z_i|\alpha^{(t)})$ is then compressed to a 20 dimensional space by computing basis $\phi_{20}(\upsilon)$ using the methods described earlier in section 3.3. The steps in this process are listed below:

1. First I define a function space (analogous to a vector space) in order to specify the domain of representation. Since I know that any prior distribution that I will need to represent must lie withing the prior space, I define the function space to be the same as the prior space.

2. Second, I compute the basis functions by drawing samples from the function space

and performing Singular Value Decomposition (SVD) on those samples. SVD gives singular vectors that form an orthonormal set, and the top singular vectors refer to the ones associated with the largest singular values. To define a 20 dimensional basis, I pick the top 20 singular vectors. Thus my basis set consists of 20 vectors or functions $\phi_{20}(\upsilon) = \{v_1, v_2 \dots v_{20}\}$, each of which is 120 dimensional. Figure 4.7 shows this basis set.

3. Now that I have defined a basis set, any distribution that lies in the function space can be expressed as a linear combination of the basis functions $\phi_{20}(\upsilon)$ using a 20 dimensional vector of coefficients $\mathbf{k} = \{k_1, k_2 \dots k_{20}\}$ as shown below:

   $p(k) = k_1 v_1 + k_2 v_2 + k_3 v_3 + ... + k_{20} v_{20}$

   These coefficients limit the space spanned by the basis to the subspace of interest. Computing the dot product of a particular basis function v_1 with the given distribution gives:

$$\begin{aligned} v_1.p(k) &= k_1 v_1.v_1 + k_2 v_2.v_1 + k_3 v_3.v_1 + ... + k_{20} v_{20}.v_1 \\ &= k_1 \quad \text{(since } \phi(\upsilon) \text{ is an orthonormal set, } v_i.v_j = 1 \text{ if } i = j \text{ and 0 otherwise)} \end{aligned} \tag{4.21}$$

   This implies that $\langle p(k)_{120}, \phi(\upsilon)_{120*20} \rangle_{20} = \mathbf{k}$, i.e., the unique set of coefficients $\mathbf{k}$ used for parameterizing the distribution of interest can be obtained by simply taking a dot product of the distribution with the set of basis functions. In other words, a dot product of the distribution with the basis set projects the high-dimensional distribution (120 dimensional in this case) to a low-dimensional space (20 dimensional in this case). This reduces the number of resources required to represent the distribution, since the number of neurons required to represent the coefficients $\mathbf{k}$ is significantly

lower than that required to represent the original 120 dimensional distribution.

4. To reconstruct the distribution from the low-dimensional space (20 dimensional) back to the high-dimensional space (120 dimensional), I make the following observation:

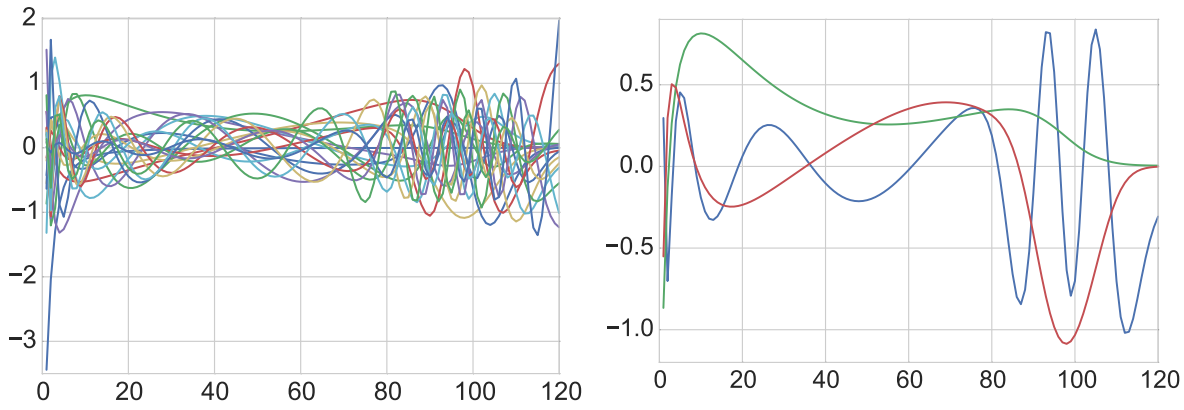
   $\langle p(k)_{120}, \phi(\upsilon)_{120*20} \rangle_{20} = \mathbf{k} \iff p(k) = \langle \mathbf{k}_{20}, transpose(\phi(\upsilon))_{20*120} \rangle_{120}$

   Since $\phi(\upsilon)$ is an orthonormal matrix, its inverse is same as its transpose. Thus, a distribution can be projected back to the high-dimensional space by simply taking a dot product with the transpose of the basis set $\phi(\upsilon)$.

5. The basis $\phi(\upsilon)$ is also used to determine the neuron encoders (as given by equation 3.18) and decoders (as given by equation 3.19) for the `cortical2` population. Specifically, the 120 dimensional function space is projected to the 20 dimensional space using the basis set. The 20 dimensional function space is then used to sample encoders and to compute the decoders.

The method outlined above provides for an efficient neural representation of the high-dimensional distribution (current optimal prior $p(z_i|\alpha(t))$) in a 20 dimensional `cortical2` population. The output from `cortical2` is reconstructed back to the 120 dimensional space, and used either along with the current or the next incoming sensory observation, x_i (in the `sensory` node), to continue the iterative process of learning the optimal prior.

Note that I project the prior distribution back to the high-dimensional space because the posterior is being computed mathematically in the `sensory` node, so there is no advantage for computing it in the compressed space. However, in chapter 5, I show how the computation to determine the posterior can be implemented in neurons. In that case, the prior (and likelihood) is not reconstructed, since computing the posterior in the compressed space is beneficial due to the significant reduction in the number of neurons required.



(a) (b)

Figure 4.7: Plots showing the basis set $\phi_{20}(\upsilon)$. (a) Plot showing the entire basis set, $\phi_{20}(\upsilon)$ i.e., all 20 basis vectors that are 120 dimensional each. (b) Plot showing three randomly selected basis functions from the set of 20 for clear visualization.

4.4.5 Results

I use a skewed Gaussian distribution (true prior for the total life spans shown in figure 4.1A) to generate the observations x_i for training my model. Each observation is allowed to cycle for three iterations before the next observation is provided. I test my model with two different prior spaces of sizes $M = 27$ and $M = 108$ and evaluate its performance by computing the root mean squared error (RMSE) of the optimal prior to which the model converges, relative to the true prior distribution. Both prior spaces contain distributions with RMSE ranging from 0 to 0.05. Figure 4.6 shows some of the distributions to which the model converges during training. This is meant to graphically clarify the implication of different RMSE values relative to the true distribution.

Figures 4.8(a, b) show the prior space of size $M = 27$. Figure 4.9a shows the results

obtained from the neural model when this prior space is used. With only one observation, the model picks the prior distributions with RMSE as high as 0.017. This implies that 25-50% of trails converge to Erlang priors instead of the true Gaussian prior. However, with just a few more observations, the model always converges to distributions with RMSE lower than 0.007 implying that it converges to some Gaussian distribution close to the optimal. On further increasing the number of observations, there is only a slight improvement.

Figure 4.9b shows the results obtained on using the prior space of size $M = 108$ shown in Figures 4.8(c, d). In this case, the initial spread of errors is even higher (upto 0.020). This is expected due to the increase in the size of the search space. The high error implies that the model picks Erlang and Power Law distributions instead of the true Gaussian prior in some trials. However, with an increase in the number of observations, the mean RMSE decreases and quickly stabilizes after around 10 observations. Interestingly, given enough observations, the model performs better with the larger prior space ($M = 108$) as compared to the smaller prior space ($M = 27$). In particular, with 10 or more observations, 50% of the trials have RMSE lower than 0.0025 and most of the rest are lower than around 0.0035. These distributions are very close to the optimal prior as can be judged from Figure 4.6. This performance improvement is due to the fact that a bigger prior space is denser and thus allows for easier navigation through the search space, by preventing the model from getting stuck in local minima. On the contrary, a smaller prior space is sparse and more prone to getting stuck in the local minima.

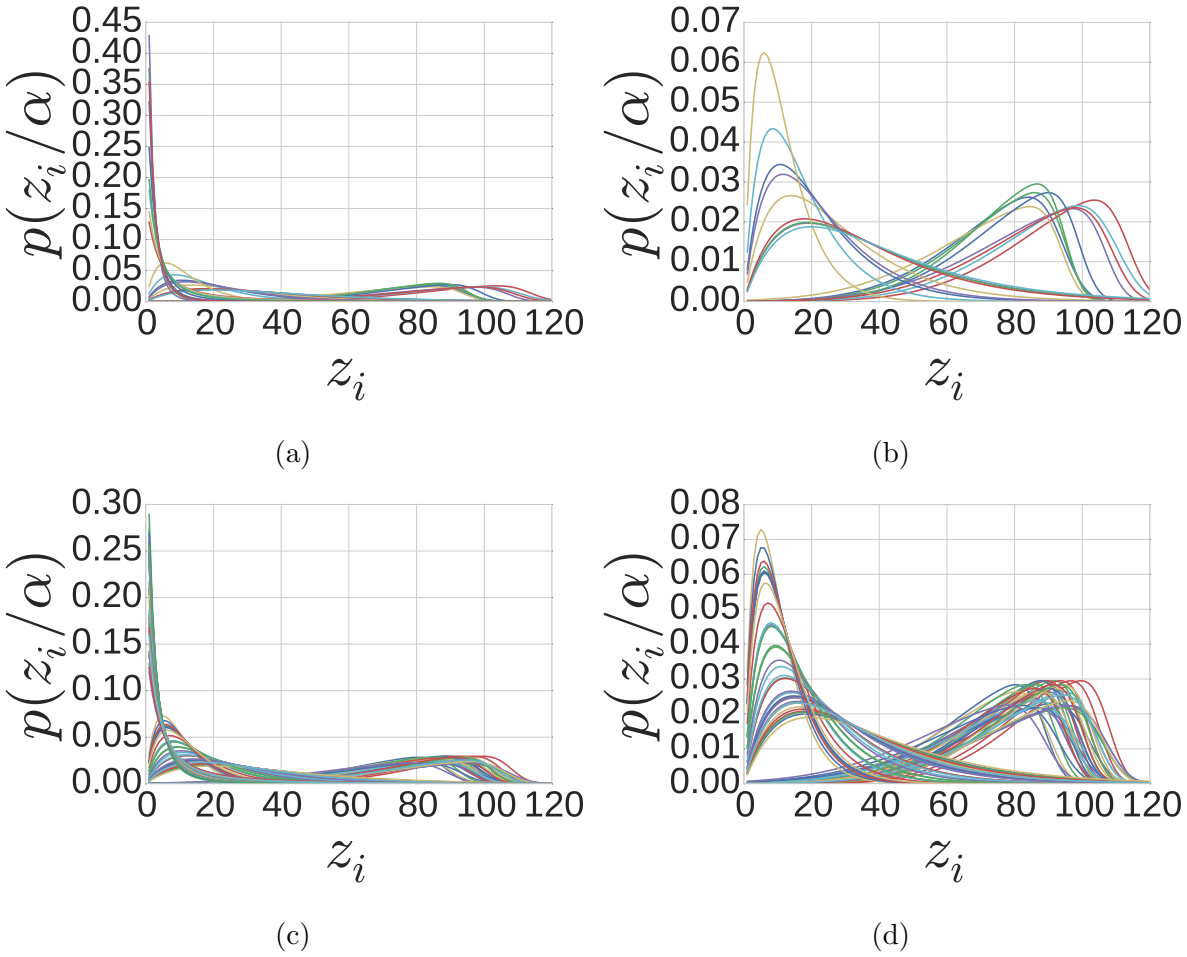
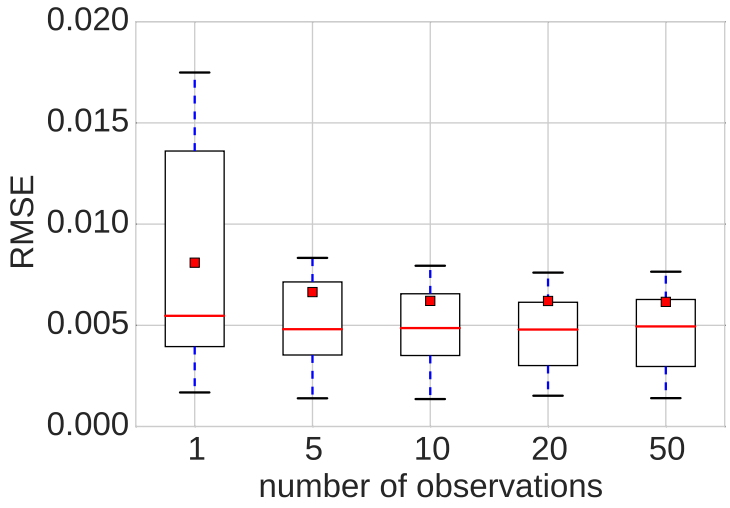
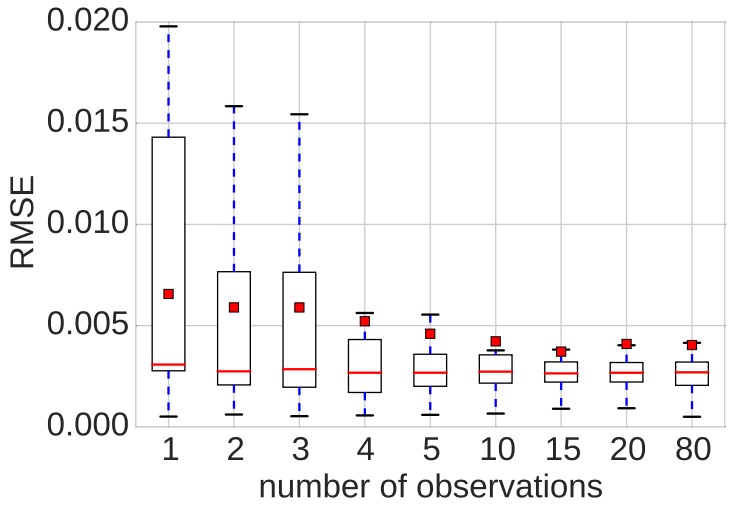


Figure 4.8: (a) Prior space with $M = 27$. (b) Prior space with $M = 27$, showing only Gaussian and Erlang families for clarity. (c) Prior space with $M = 108$. (d) Prior space with $M = 108$, showing only Gaussian and Erlang families for clarity.



(a)



(b)

Figure 4.9: Results from the neural model when a skewed Gaussian distribution is the true prior. The y-axis shows the root mean squared error relative to the true prior and the x-axis shows the number of observations used for training the model. The red squares show the mean and the red bar shows the median of the RMSE data from 100 trials.(a) Prior space of size $M = 27$. (b) Prior space of size $M = 108$.

However, an increase in the prior space size also leads to an increase in the number of neurons required for the model. In general, number of neurons scale linearly with prior space size M, and are given by: $460 \times M + 200$. One area for future work is to explore whether this scaling can be further improved by applying the same methods of dimensionality reduction that I use to represent the optimal prior in `cortical2` ensemble in my model. The M dimensional utilities represented by `cortical1` ensemble can be treated as a high-dimensional vector and a basis space can be computed to compress it to a low-dimensional space. This would mean that the number of neurons required for `cortical1` ensemble and the Memory network would be significantly reduced, similar to `cortical2`. This compressed representation can then be reconstructed before input to the WTA network. Additionally, the prior space $p(z_i|A)$ and its log function are $120 \times M$ dimensional matrices stored in the weight matrices at two different connections in the network. This encoding can also be made more efficient by compressing this matrix to a low-dimensional space using the same basis set. Both these issues are discussed in detail in section 6.3.

4.5 Summary

In this chapter, I proposed neural mechanisms that may be used by humans for learning priors from experiences of the natural world. The neural model built to accomplish this is based on two assumptions: (1) The likelihood assumption from Griffiths and Tenenbaum (2006) which is given by equation 4.3, and (2) Prior space assumption explained in section 4.4.1. Both these assumptions and their implications are discussed again in section 6.4. The model is capable of learning priors for any task that satisfies these assumptions. I also showed how the beliefs or priors can be represented in the brain during and after the learn-

ing process using the methods outlined in section 3.3. Specifically, the computation of an orthonormal basis set $\phi(v)$, compression of a probability distribution to a low-dimensional space, and its reconstruction back to a high-dimensional space were described in detail.

Through the process of building this model, I illustrated a systematic approach towards building a neural model, that integrates different levels of analysis presented in section 2.3. I started with the problem description at the abstract computational level in section 4.3, made assumptions at the learning and evolution levels in section 4.4.1 and finally built the model at the algorithmic level, where I proposed neural mechanisms constrained by implementation level details in section 4.4.4. This process demonstrates the interdependence of these levels.

The model can be used to learn priors for a variety of psychological tasks, and I evaluated its performance by training it on a life span inference task with the goal of learning the prior on the life spans of humans. The results showed that for a prior space of size $M = 108$, the model converges to the optimal or near optimal priors after being trained with only around 10 observations. Though the model is able to successfully learn the prior, it is not clear how these priors may be used for making inferences, and how the computations to do so might be realized in the human brain. This is the question that I address in the next chapter.

Chapter 5

A Spiking Neural Model for Bayesian Inference

5.1 Introduction

In the previous chapter, we saw a neural model that can be used to learn priors for a wide variety of inference tasks. I demonstrated how the model can learn the prior for the life span inference task. However, to show that the neural representation of the learned prior can be used to perform inference, I use this prior to infer life spans and then evaluate the accuracy of the inference against human predictions.

In this chapter, I present a spiking neural model that is designed to perform Bayesian Inference in a neurally plausible way. The model is general purpose and can be used to infer a posterior distribution given a likelihood and a prior. The model is demonstrated using the life span inference task. It uses the prior over the total life spans learned by the previous model (presented in chapter 4) to infer the life spans of individuals given

their current age. The predictions made by the neural model are then compared to human predictions on this task. I show that the neural model matches human performance on this task better than an ideal Bayesian model due to the use of neuron tuning curves.

5.2 Neural Model

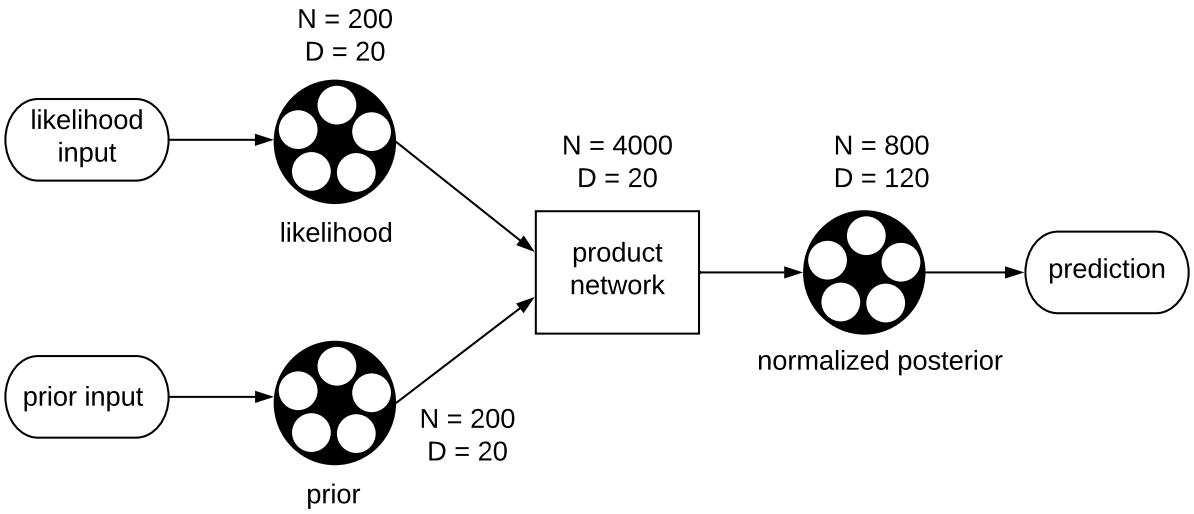


Figure 5.1: A schematic diagram of the neural model for Bayesian inference. The circles represent neural populations. The rectangles group the neural populations into subnetworks. The rounded boxes represent nodes (non-neural). The flow of information is denoted with arrows ($\rightarrow$). The number of neurons and dimensions represented by each population/network are labelled as N and D respectively.

Figure 5.1 shows the architecture of the neural model designed for performing general Bayesian inference. In other words, given a prior and a likelihood, the neural model can

combine both of them to compute a posterior distribution in accordance with the Bayes' rule given by equation 4.1. Similar to the previous model, this model is also built using the Nengo neural simulator based on the NEF.

In this section, I explain how this model can be used to infer life spans. Recall that in the previous model presented in section 4.4.4, the `sensory` node used the prior from the previous iteration, computed the likelihood based on the current observation x_i and used both these quantities to compute the posterior distribution $p(z_i|x_i, \alpha^{(t)})$. Importantly, the posterior distribution was computed mathematically. For this model, I assume that the prior distribution and the likelihood function are given from the previous model, and show how the posterior distribution can be computed in a neurally plausible way. First, I represent the given prior and likelihood distributions in neural populations. Next I use the product network to compute the posterior distribution through an element-wise product of the prior and likelihood. Once I have the posterior distribution, I use a prediction function to obtain a prediction for the total life span from the posterior.

5.2.1 Product Network

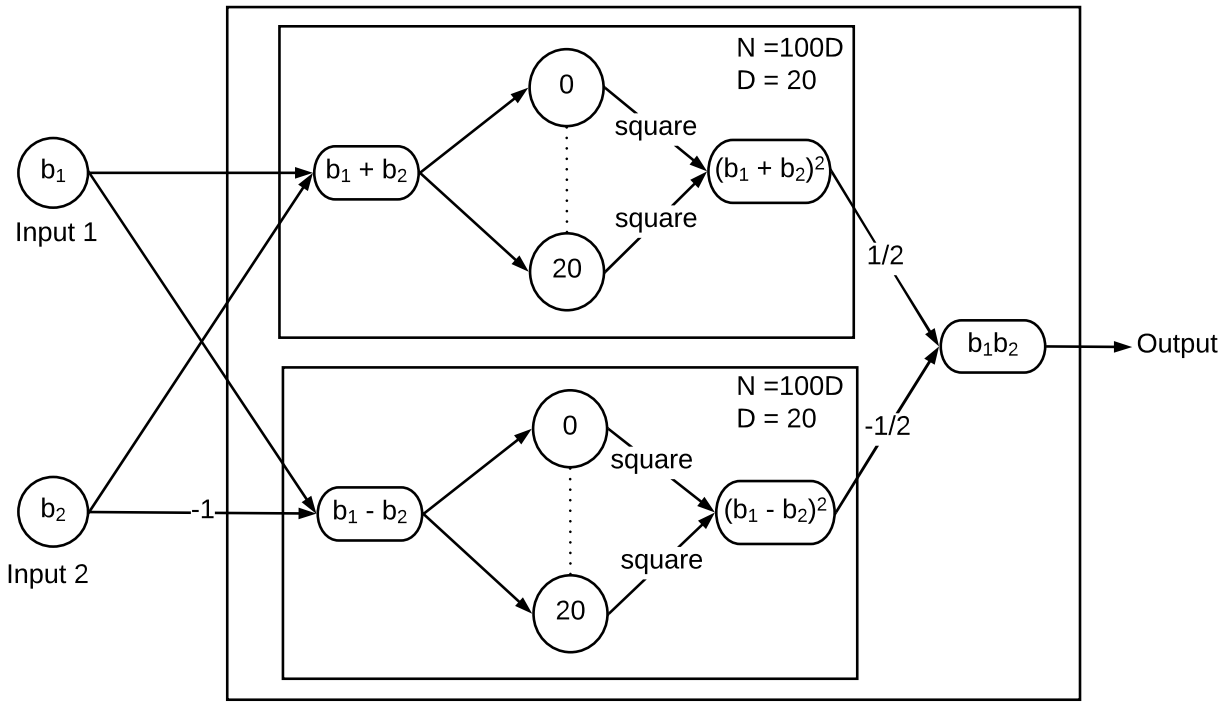


Figure 5.2: A schematic diagram of the Product Network. The circles represent neural populations. The rectangles group the neural populations into subnetworks. The rounded boxes represent nodes (non-neural). The flow of information is denoted with arrows ($\rightarrow$). N and D respectively indicate the number of neurons and dimensions in each sub-network. Note that the non neural elements are shown here only for ease of understanding, and the final simulated model does not have any non neural elements. It consists of only spikes propagating through the connections, and connection weights.

Though multiplication is nonlinear, it has a well-characterized implementation in neurons that does not require nonlinear interactions, and can be implemented accurately with the

NEF. The product network makes use of this characterization. It computes the element-wise product of two equally sized vector inputs using diagonal encoders (Gosmann, 2015). For instance, given inputs b_1 and b_2 which are 20 dimensional each, the 20 dimensional product $b_1 b_2$ can be computed as follows:

$$\begin{aligned} b_1 b_2 &= 1/4(b_1^2 + 2b_1 b_2 + b_2^2) - 1/4(b_1^2 - 2b_1 b_2 + b_2^2) \\ &= 1/4(b_1 + b_2)^2 - 1/4(b_1 - b_2)^2 \\ &= 1/4((1,1)^T.\mathbf{b})^2 - 1/4((1,-1)^T.\mathbf{b})^2 \\ &= 1/2[(\mathbf{e_1}.\mathbf{b})^2 - (\mathbf{e_2}.\mathbf{b})^2] \end{aligned} \tag{5.1}$$

where $\mathbf{e_1}$ and $\mathbf{e_2}$ are encoders (unit vectors) given by $\mathbf{e_1} = (1/\sqrt{2}, 1/\sqrt{2})$ and $\mathbf{e_2} = (1/\sqrt{2}, -1/\sqrt{2})$.

Figure 5.2 shows a schematic diagram of the product network. It consists of two subnetworks that compute the first and second square terms respectively in equation. 5.1. These networks consist of 20 neural populations (one for representing each dimension). While one of these networks receives the sum of inputs $b_1 + b_2$, the other receives the difference of inputs $b_1 - b_2$ (due to the negative transform computed at the connection from input b_2). Each neural population (from 0 to 20) receives one dimension as input from the input nodes. A square function is computed at the output connection from each of these neural populations using the principle 2 of the NEF. Thus the output from the two sub-networks is $(b_1 + b_2)^2$ and $(b_1 - b_2)^2$ respectively. Finally, the connection from the output nodes of these sub-networks to the output node of the product network (which is the outer network containing the sub-networks) contains diagonal encoders $\mathbf{e_1}$ and $\mathbf{e_2}$. This connection computes a transform of 0.5, thus giving the product $b_1 b_2$ as the output.

5.2.2 Model Implementation

To represent the prior and likelihood in neural populations, I again use the methods described in section 3.3, following a process similar to `cortical2` in the previous model (section 4.4.4). Specifically, I define a basis $\phi_{20}(\upsilon)$ to span the space of each distribution. To compute the basis I sample from a family of 120 dimensional distributions and do Singular Value Decomposition to obtain a 20 dimensional basis. This basis is used to determine the encoders (as given by equation 3.18) used in the NEF simulation. The same basis is used for the optimization to find the neuron decoders (as given by equation 3.19) that are needed to perform the desired computations. Similar to the encoding and decoding functions, the 120 dimensional prior and likelihood functions are also projected to the 20 dimensional space through weights over the basis. Note that separate bases $\phi_{20}(\upsilon)^{prior}$ and $\phi_{20}(\upsilon)^{likelihood}$ are computed for compressing the prior and likelihood respectively to the 20 dimensional space.

In figure 5.1, the `likelihood input` and `prior input` are nodes that provide the named 20 dimensional inputs to the neural populations `likelihood` and `prior` respectively. Both `likelihood` and `prior` populations are 20 dimensional and contain 200 LIF neurons each. The product network receives input from these populations and computes the posterior distribution (in the 20 dimensional space). It does so by computing an element-wise product of its inputs as explained in section 5.2.1. The product network contains 40 neural ensembles of 100 neurons each for a total of 4,000 neurons.

The output connection from product network to `normalized posterior` reconstructs the posterior back to 120 dimensional space and computes the normalization function using principle 2 of the NEF. The posterior is reconstructed using the basis $\phi_{20}(\upsilon)^{posterior}$ that is computed by taking the element-wise product of $\phi_{20}(\upsilon)^{prior}$ and $\phi_{20}(\upsilon)^{liklihood}$. The

reconstructed normalized posterior distribution is represented by `normalized posterior` population that is 120 dimensional and contains 800 LIF neurons. Next I compute the median of this distribution on the connection between the `normalized posterior` population and the `prediction` node (using principle 2 of the NEF). I read out the model prediction from the *prediction* node.

5.3 Results

5.3.1 Connecting the two models

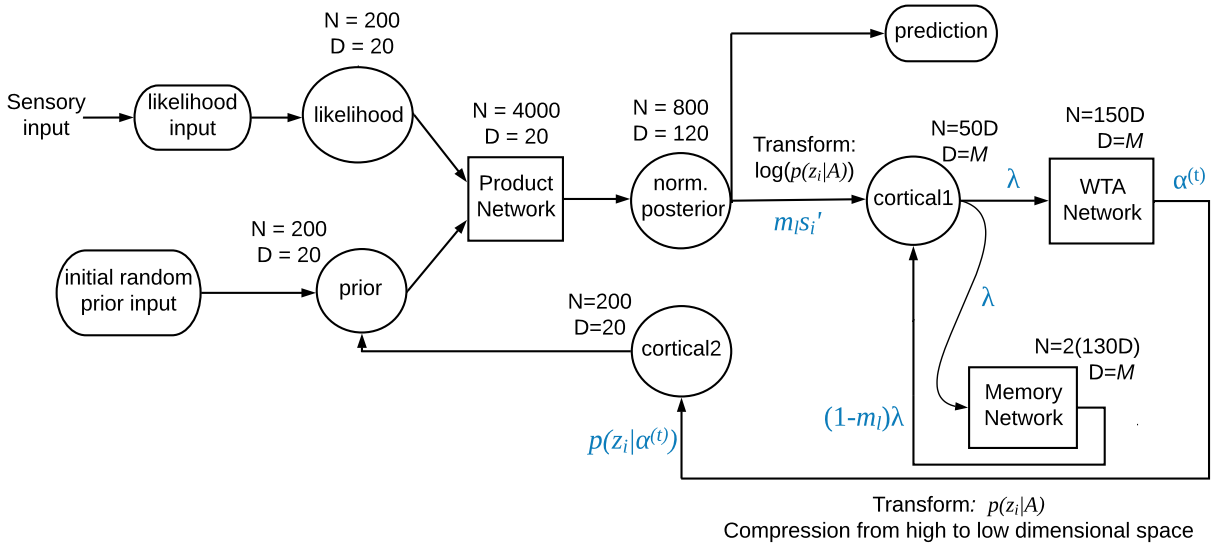


Figure 5.3: A schematic diagram of the complete neural model for learning priors and inferring from them. The circles represent neural populations. The rectangles group the neural populations into subnetworks. The rounded boxes represent nodes (non-neural). The flow of information is denoted with arrows (→).Transformations are labelled along these connections. The number of neurons and dimensions represented by each population/network are labelled as N and D respectively. The blue labels indicate the quantity encoded by the spike trains propagating through the connections.

In this section, I first show how the model presented in this chapter connects to the one presented in the previous chapter. Figure 5.3 shows the schematic diagram of the complete

neural model. The `sensory` node in the model for learning priors (figure 4.5) has been replaced by the inference model (figure 5.1). As mentioned before in sections 4.3 and 4.4.1, this model can be used to learn the priors for any task in which the upper limit of a duration, extent or any other numerical quantity needs to be predicted, given a sample drawn from that interval. As the prior for a particular task is being learned, it can also be used simultaneously to make predictions.

I use this model to obtain results for the life span inference task. In the model in figure 5.3, the sensory input provides an observed current age x_i that is used by the `likelihood input` node to compute the likelihood function based on equation 4.19. This 120 dimensional distribution is then compressed to the 20 dimensional space on the connection to `likelihood` population, which represents the likelihood in the compressed space.

The model begins with a random initial guess at the hyperparameters $\alpha^{(t)}$ of the prior, i.e., a randomly chosen prior from the prior space (section 4.4.1) is provided as input. Similar to the likelihood, the initial prior is also compressed to the 20 dimensional space and represented by `prior` population. The model then computes the normalized posterior as explained before in section 5.2.2, which is represented by the `norm. posterior` population. Recall that I earlier referred to the normalized posterior as $T(x_i, z_i)$ (equation 4.13). Thus the output of the `norm. posterior` population is reconstructed and multiplied by m_l through a node (not shown in the figure) to give $T(x_i, z_i) * m_l$. This was the output from the `sensory` node in figure 4.5. The rest of the model works as explained in section 4.4.4, to learn the prior iteratively. Subsequent to the initial input, the input to the `prior` population comes from `cortical2`, which represents the current optimal prior $p(z_i|\alpha^{(t)})$ in the 20 dimensional space.

The other output of `norm. posterior` is used to get the prediction out of the model as explained in section 5.2.2. The prediction is based on the current age x_i and it is possible to

look at the predictions given different values of x_i while the optimal prior is being learned. As the learning proceeds, the accuracy of predictions improves over time. Once the optimal prior has been learned and stabilized, we can provide the desired current age x_i as input to the model and read out the predicted life span from the `prediction` node. This is how I obtain the results presented in the next section.

5.3.2 Comparison to human behavioral data

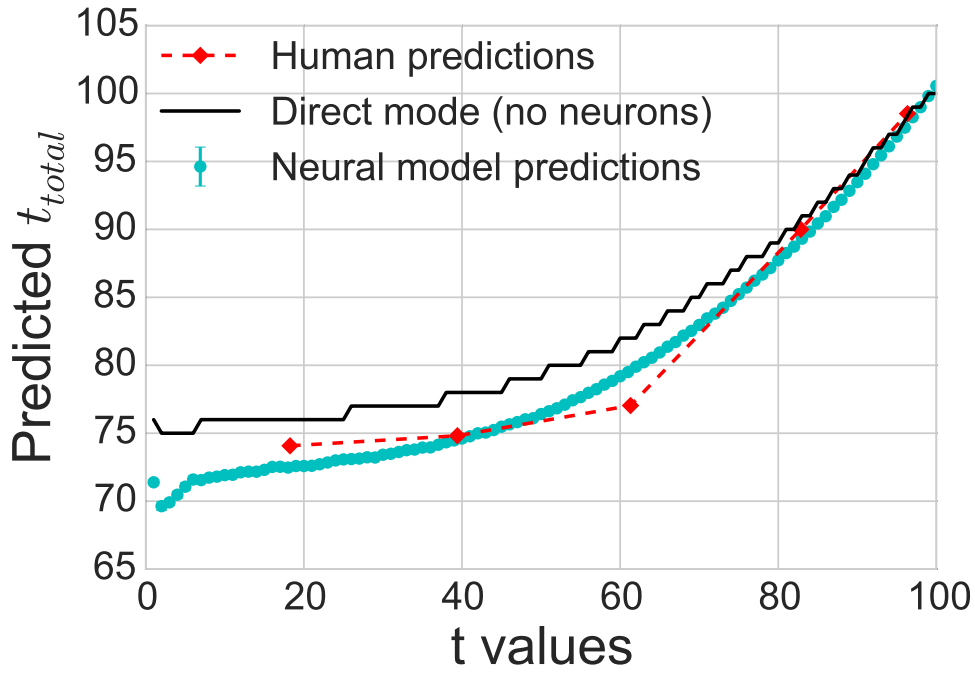


Figure 5.4: Inference results from the neural model (95% confidence intervals), compared to humans and Direct mode - our model with computations in low-dimensional (20 dimensional basis) space, but without neurons.

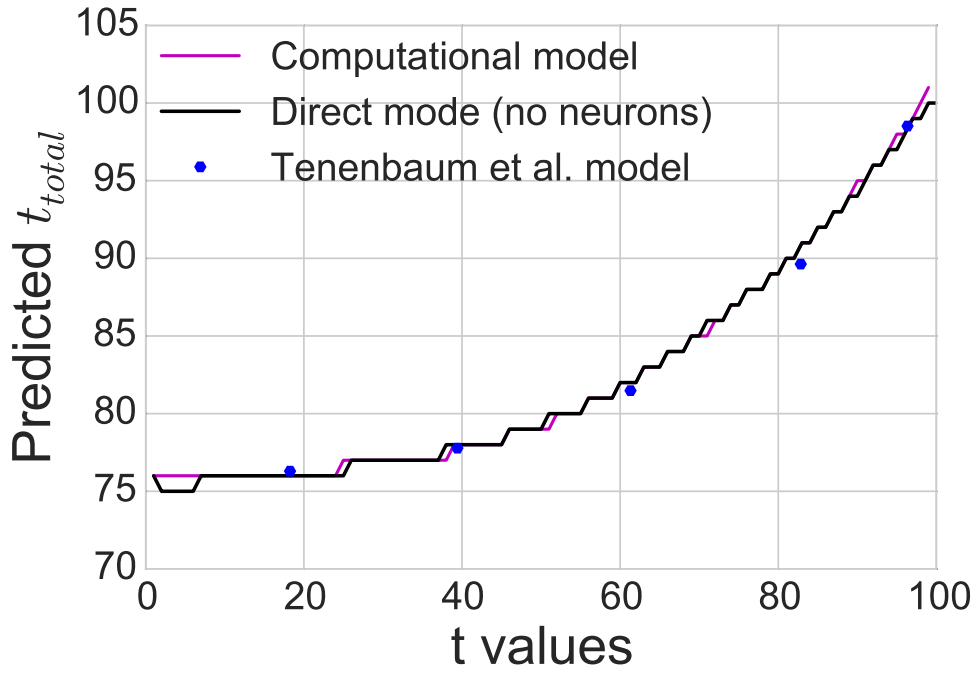


Figure 5.5: Inference results from Griffiths and Tenenbaum (2006) model (only data corresponding to human data), Computational model i.e., our replication of their model, and Direct mode i.e., our model with computations in low-dimensional space, but without neurons. The plot shows that there is no error caused due to the low dimensional embedding.

Recall from section 4.3, that each x_i corresponds to the current/observed age t of an individual, and each z_i corresponds to the unknown life span t_{total} of the same individual.

Figure 5.4 shows the inference results obtained from the neural model. Model predictions are plotted for current ages t from 1 to 100. Human predictions are plotted for five data points. They were obtained from figure 4.1B using a web plot digitizer. Neural model predictions along with the predictions in direct mode are also plotted. In direct mode, the model doesn't simulate neurons, and all the computations in the model are performed

mathematically. However, the dimensionality reduction is still carried out and the computations are performed using the low dimensional (20 dimensional) distributions. The difference between the results in Direct mode and Neuron mode is due to the limited number of neurons in the `normalized posterior` population. As the number of neurons in this ensemble increases, the results approach the Direct mode results (800 neurons provide the best fit to human data). Thus, neural results match the human data better due to the approximate representation of the normalized posterior by the neurons in the `normalized posterior` ensemble.

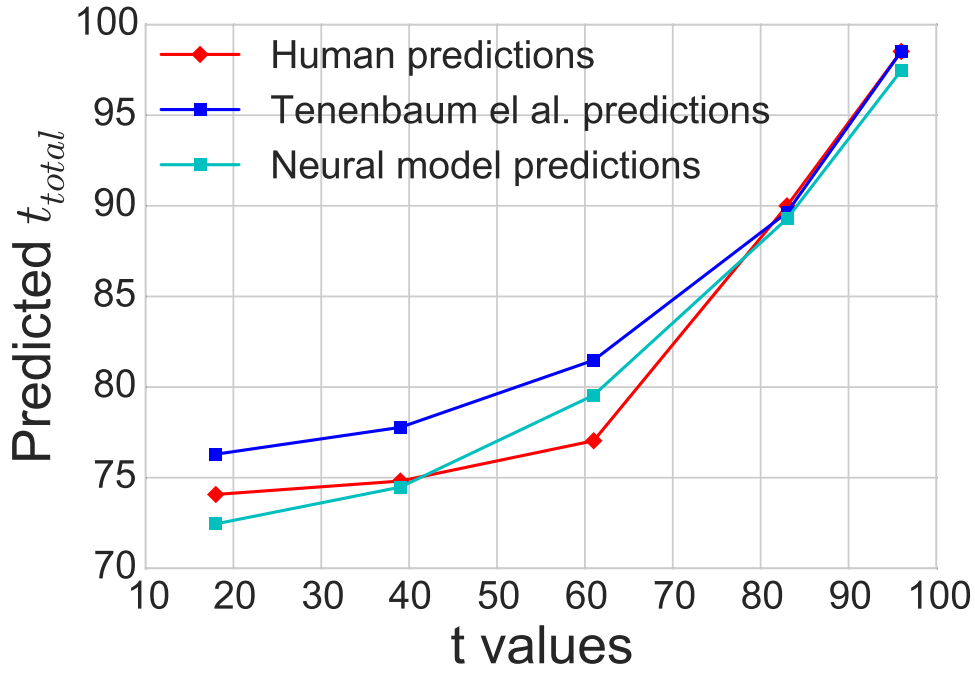


Figure 5.6: Kolmogorov-Smirnov (K-S) test results. Dissimilarity relative to human predictions - Griffiths and Tenenbaum (2006) model: 9.628, neural model: 1.959. Neural model and human data are median predictions.

Since the model performs computations using low-dimensional distributions, it is possible that the compression to a low-dimensional space may introduce some error in the predictions, thus causing them to deviate from the optimal Bayesian predictions. To verify this, predictions from the Griffiths and Tenenbaum (2006) model, the computational model (our replication of their model), and direct mode (our model with computations in a compressed 20 dimensional space, but without neurons) were compared. The results of this comparison are shown in Figure 5.5. Since the results obtained from the direct mode are the same as the computational model, the low-dimensional embedding is not losing any information and thus does not introduce any error. However, I expect some error due to this constraint for more complex distributions.

I use the Kolmogorov-Smirnov (K-S) test to examine the goodness of fit of the neural model predictions relative to the Griffiths and Tenenbaum (2006) model. The data used for the K-S test is shown in Figure 5.6. The dissimilarity of the Griffiths and Tenenbaum (2006) model relative to human predictions is 9.628, while that of the neural model is 1.959, indicating the much closer fit of the neural model to the human data.

Overall, my results suggest that the closer fit of the neural data can be solely attributed to fitting the neuron tuning curves in the *norm. posterior* population, where 800 neurons provide the best match to human performance. Since the low-dimensional neural implementation can be made to match the human data, this is some evidence in support of the hypothesis that human brains represent low-dimensional state spaces (low-dimensional parameterizations of high-dimensional distributions fit using neural tuning curves).

5.4 Summary

In this chapter, I proposed neural mechanisms for performing Bayesian inference. First a neural model was presented that combines neural representations of learned priors with the neural representations of the likelihood (computed based on new observations) to perform Bayesian inference in a low-dimensional space. It computes a posterior distribution using a product network that performs an element-wise multiplication of its two inputs (prior and likelihood). The posterior is then reconstructed back to a high-dimensional space and its median is computed to get the prediction from the model.

Next, this model was connected to the model presented in chapter 4, such that it can be used in the process of learning priors and also enable the learned priors to be directly used for making predictions. The complete model was trained on a life span prediction task and the predictions based on the prior learned by the model were compared to: (1) human predictions, (2) the computational Bayesian model from (Griffiths and Tenenbaum, 2006), and (3) the non-neural version of my model, i.e., the direct mode that doesn't simulate neurons but still uses the low dimensional embedding of the distributions. The results showed: (1) a close match between the predictions from the neural model and those from humans, (2) a better fit of the neural model to human predictions relative to the computational Bayesian model, and (3) no error caused by the process of dimensionality reduction in the neural model.

Since the results from the neural model match human predictions, this provides some evidence in support of the neural mechanisms proposed both for learning and inference, and the hypothesis that human brains may represent functions embedded in a low-dimensional space. However, more experiments need to be conducted in order to further validate/refute the model. Furthermore, there are certain things in the model that can be improved to

make it more computationally efficient, and provide room for future work. In the next chapter, I discuss some key aspects of the model that haven't been discussed in detail yet, and provides avenues for future work.

Chapter 6

Discussion and conclusions

In this chapter, I focus on discussing some topics related to the neural model presented in the previous chapter. The discussion includes some of the limitations of the neural model, and suggests avenues for future work both in experimental and theoretical domains. Some of the sections also clarify the claims being made in this thesis. This is followed by the final section that concludes this thesis by revisiting the goals I started with in section 1.2.

6.1 Testing the neural model

As we saw in section 5.3, one way of testing the model is to evaluate its performance relative to behavioral data. In section 5.3, I show that it is possible to run both the prior optimization and inference at the same time, and both become progressively more accurate over time. I then show that the model predictions based on learned priors provide a close match to the human behavioral data. The data was obtained from the behavioral experiment conducted by Griffiths and Tenenbaum (2006) (refer to section 4.2.1). The

subjects in the experiment were undergraduate students, i.e., people between ages of 18 to 23. This implies that their predictions were based on the learned optimal priors, since by this age people have presumably already learned the distribution over the total life spans. They know that the average life span is around 75 years (variable based on geographical location) and that a 100 year old person is more likely to die sooner than a 40 year old.

Though the available behavioral data provides evidence in support of the mechanisms used in the neural model, additional behavioral data might be useful to validate the learning procedure used. For instance, longitudinal studies over the development period of children can help provide data that may reflect how the predictions of children change over the course of learning. For instance, children can be asked to perform the life span inference task repeatedly at different stages of their life with a gap of several months to an year. Their predictions can be used to compare the predictions of the model during the learning phase, and provide evidence to further support or refute the neural mechanisms for learning priors proposed in section 4.4.4. Furthermore, children may develop biases depending on the environment they grow up in. For instance, average life expectancy in India is 68 years while that in Canada is 82 years. Hence, children in these countries are likely to have different priors over the total life spans. Thus, collecting behavioral data from longitudinal studies across different populations and then comparing it to the predictions of the model when trained on similar data can be a useful approach towards testing the model, as well as testing and refining the hypothesized prior space used in the model.

6.2 Brain areas

Though behavioral data can be used to validate models on the computational and even algorithmic levels, it doesn't tell us much about the implementation level. Which areas

of the brain may be involved in learning and inference tasks like the ones we are talking about? What is the connectivity map among these areas? Which neural populations in these brain regions should we record from to compare the activity of the neural populations from our model? These are challenging questions given our limited understanding of the brain.

It is well-known that prefrontal and frontal cortices are the most likely regions of the brain involved in any kind of high level reasoning and inference (Donoso et al. 2014, Christoff et al. 2001). However, the brain is a complex machine, and often makes use of multiple regions to accomplish a specific task. Hence, the implementation level questions are much harder to answer and require experimental data from fMRI (functional magnetic resonance imaging) and electrophysiology studies. Specifically, fMRI studies have been found to be very useful in determining the functional neuroanatomy. Behavioral studies described in the previous section could be conducted in an MRI machine in order to determine the brain regions involved in the inference task. Furthermore non-invasive electrophysiology techniques like EEG (electroencephalogram) and MEG (magnetoencephalography) can also be used for mapping brain activity during behavioral studies. Such techniques can inform our understanding of how the learning and inference might be happening at the implementation level in the brain, and form a useful area for future work.

6.3 Encoding of the prior space

The proposed network architecture assumes a set of priors that are already stored in the weights of the network. In the model in figure 5.3, the prior space is stored in the weights of two connections: (i) $log(p(z_i|A))$ on the connection between the `norm. posterior` and `cortical1`, and (ii) $p(z_i|A)$ on the connection between WTA network output and

`cortical2`. These stored priors can be potentially encoded in a more efficient manner. For instance, instead of storing the full $120 \times M$ dimensional prior space matrix, the distributions in the prior space can also be compressed similar to other distributions that were compressed and represented in the neural populations. The same basis space $\phi_{20}(\upsilon)$ (as computed for `cortical2` in section 4.4.4) can be used for projecting the prior space to a 20 dimensional space. This would reduce the weight matrix size to $20 \times M$. However, this would also imply that the reconstruction of the normalized posterior back to 120 dimensional space wouldn't be required at the connection between the `norm. posterior` population and the WTA network. This reduces the dimensionality of `norm. posterior` from 120 to 20 thereby reducing the number of neurons. This hasn't been tested yet, and is an important area for future work.

Additionally, as mentioned in section 4.4.5, an increase in the prior space size M is another factor that contributes to an increase in the number of neurons in the model. The methods of dimensionality reduction can also be applied to the prior space size, however this is a tougher problem to solve. Recall that the prior space size M refers to the number of distributions assumed to be in the prior space. One way to reduce this number is to pick a set of distributions that can be used as a basis set to generate all the distributions in the prior space through their linear combination. For instance, a set of skewed Gaussian distributions could be used such that their linear combinations lead to a variety of different distributions. However there are a few challenges that come with this approach:

- Choosing a set of basis functions to generate all the distributions in the prior space can be a challenge, especially since different distributions in the prior space belong to different families.
- Assuming that we are able to compress M down to K (where $K << M$) basis

functions, it is not clear how the competition between the K functions at the WTA stage of the network could be interpreted in terms of the original M distributions.

Though it is a challenge to compress M, if accomplished it would not only further reduce the size of the weight matrices encoding prior space from $M \times 20$ to $K \times 20$, but will also lead to a reduction in the number of neurons required for `cortical1` and the memory network, similar to `cortical2`. Therefore, this also forms an important area to be researched in the future.

6.4 Scalability to other inference tasks

The neural model presented in this thesis can be used for learning priors and making inferences for any task where the upper limit of any quantity needs to be determined given a sample from that interval. However, the model rests on the following assumptions:

1. Likelihood assumption: The training data is generated using a hierarchical Bayesian network (section 4.4.2) that is based on the assumption that there is an equal probability of observing any sample from the given interval i.e., $x_i|z_i \sim \mathcal{U}[0, z_i)$ and the likelihood function used in the neural model for such inference tasks is given by the equation 4.19.

2. Prior space assumption: The model assumes a prior space (section 4.4.4) based on psychological data from Griffiths and Tenenbaum (2006).

Therefore, the model can only scale to the tasks that follow assumption 1 and have priors of the form assumed in 2. However, assumption 2 can be removed or at least relaxed

if a suitable basis set of a small size K is used, such that any required prior distribution can be constructed through the linear combination of these K distributions as suggested in section 6.3.

The likelihood assumption is relevant at three places:

- In the hierarchical Bayesian network used for generating the training data. This network is not a part of the neural model, and hence can easily be changed to adapt to any task at hand. In other words, the neural model is agnostic to how the data used to train it is generated. For instance, we could even use the data recorded from the real world to train the model.

- In the `likelihood input` node shown in figure 5.3. Here the likelihood is computed based on the current observation x_i. However, since this is a node, the likelihood is computed mathematically and compressed to a low dimensional (20 dimensional) space, before it is sent to the `likelihood` population. The `likelihood input` node can be programmed to compute any likelihood function irrespective of its form.

- In the weight matrix on the connection between `norm. posterior` and `cortical1` in figure 5.3. The weights on this connection are independent of the likelihood since the likelihood is usually a function of z_i as is the case in equation 4.19. As explained in section 4.4.4: equation 4.20, the dot product of the likelihood with the posterior gives a scalar value that doesn't affect the relative values corresponding to the M competing priors.

Based on the above points, only the computation in the `likelihood input` node is affected by the likelihood assumption. However, note that the `likelihood input` node is not really a part of the neural model, i.e., it computes the likelihood mathematically and

I do not propose any neural mechanisms for doing so. Care must be taken to keep the scalability of the model in mind, when considering neural mechanisms for the likelihood computation. The rest of the neural model is scalable to the tasks having a likelihood that's either constant or expressed as a function of z_i, which is usually the case.

6.5 Basis function computation

Recall that in sections 3.3 and 4.4.4, I propose that we can compute basis functions to represent probability distributions in neural populations in an efficient way. Specifically, the basis functions can be used to project high-dimensional distributions to low-dimensional spaces, which require fewer neural resources for representation. These basis functions are computed using Singular Value Decomposition (SVD). This is done mathematically, and I do not make any claims about neural plausibility of the process of generating these basis functions. It is worth noting that if all computations in the model are carried out in the compressed space (as proposed in section 6.3), then we never need to know the basis functions for model implementation, except when compressing the model inputs from high to low-dimensional space.

Nevertheless, methods for doing dimensionality reduction in a neurally plausible way have been proposed before. For instance, Tripp (2009) (chapter 8) showed that Hebbian plasticity can change the dimension of the space coded by a neural population. Specifically, lateral connections within a neural population can modulate the dimension of the neural code, since they directly influence the neurons' tuning curves through their preferred direction vectors. Tripp (2009) extended Oja's original model (Oja, 1982) to show that excitatory lateral connections in a neural population reduce the dimension of the neural code, and inhibitory lateral connections increase the dimension. This is because the lateral

interactions incline the tuning curves of neurons either towards (excitatory) or away (inhibitory) from those of other neurons laterally connected to it. Mutual excitation causes the neurons to respond strongly to the same input. As a result, Hebbian plasticity draws the (feedforward) weight vectors together thus inclining the tuning curves of neurons towards each other. On the other hand, mutual inhibition prevents neurons from responding strongly to the same input, as a result of which Hebbian plasticity draws the weight vectors apart, inclining the tuning curves of neurons away from each other.

To summarize, lateral interactions can influence the dimension of the feedforward weights and hence the dimension of the post-synaptic neural code. Hebbian plasticity in combination with excitatory lateral connections in a neural population provide a possible neural mechanism for dimensionality reduction, and can potentially be used for compressing high-dimensional distributions to a lower-dimensional space in our model. However, this hasn't been implemented and tested yet, and needs further exploration.

6.6 Is the human brain really Bayesian?

I don't claim that the human brain is Bayesian in the strictest sense. The performance of our neural model doesn't match the optimal Bayesian curve, but is closer to the human-level performance. Recall from section 2.6 that one of the open questions in the field of neurally plausible Bayesian computations is whether these computations are optimal. The results from my model suggest that they are probably suboptimal when realized in the brain due to the following reasons:

1. Limited resources: Figure 5.6 shows that the neural model provides a better match to the human predictions than an ideal Bayesian model by Griffiths and Tenenbaum

(2006). This is due to fitting of the neural tuning curves in the neural populations. Given a large number of neurons, the neural model is able to replicate the ideal Bayesian results. Thus, I predict that the suboptimal performance results from limited neural resources in the brain.

2. Improperly learned beliefs: Second source of sub-optimality is the prior when used during the learning phase. If the predictions are obtained from the neural model using the incompletely learned priors, that introduces another source of error in the predictions. In this case, the predictions won't provide a good match to the human behavioral data. However, as mentioned in section 6.1, developmental studies might provide human data that can be fairly compared to the neural model predictions during the learning phase.

3. Approximate computations: Third source of sub-optimality is the dimensionality reduction of probability distributions for their efficient representation in neural populations. Though figure 5.5 shows that the low-dimensional embedding doesn't cause any error in the life span inference task, I expect that it can be a source of error if the distributions to be represented are more complex.

One can only claim that the brain is Bayesian if one believes that a "general-purpose theory of cognition" exists. However, given that our brain is arguably the most complex machine known to us as of today, and given our limited understanding of it, it is yet to be determined whether the goal of finding such a general-purpose theory to explain the brain even makes sense. Nonetheless, calling the brain Bayesian would require answering many foundational questions: How well can Bayesian theories account for irrationality in human cognition? Does the brain actually use Bayesian rules? Does it do so optimally or through approximate descriptions? Do Bayesian theories require an implausibly uniform view of

the mind? What kind of prior knowledge do we bring to bear on a particular learning or inference task? How does that knowledge interact with the examples we observe to guide our generalizations?

I haven't attempted to answer all of these questions and do not intend to make the claim that the brain is Bayesian. However, I believe that that certain computations in the brain may be Bayesian (though probably not optimal), and I have shown their neural plausibility by proposing neural mechanisms for them, and comparing the results to human predictions.

In section 2.1, I mentioned that besides the Bayes' theorem, the Bayesian Framework has been expanded to include other methods for approximate inference that are out of the scope of this thesis, e.g., variational Bayesian methods and Markov chain Monte Carlo methods. Statisticians commonly use these approximate inference methods to deal with very high dimensional problems, usually at a high computational cost. However, whether these approximate methods are neurally plausible, and whether the human brain employs them, are open questions. In this thesis, I do not make any claims about the neural plausibility of these methods.

Furthermore, I do not claim that humans are rational. There is a large body of literature on human decision making which suggests that humans do not make decisions according to 'rational' rules and do not follow models based on the utility theory (Mongin, 1997). For instance, the Ellsberg paradox (Segal, 1987) explains why most humans don't take risks and settle for the mediocre i.e., humans exhibit a strong aversion to ambiguity and uncertainty, and have an inherent preference for the known over the unknown. In the face of ambiguity in the probabilities of the outcomes, humans may formulate a vague probability of possible outcomes which may be biased. For example, most humans avoid risk, and many assume that if they are not told the probability of a certain event in a real-life scenario,

it is to deceive them. This and other assumptions may affect their decisions, and hence their decisions may not follow models based on classical probability. Thus, I suggest that the probability distributions that humans represent may not always be accurate i.e., they may be biased. However in this thesis, I have given a neurally plausible account of how humans may represent probability distributions, regardless of whether these probabilities are accurate or biased, and how these representations can be used in meaningful ways to perform inference in a human-like way (which may not be optimal).

6.7 Conclusions

This thesis aimed to explore the neural plausibility of Bayesian inference. I proposed neural mechanisms for two important components of Bayesian inference: (1) learning priors, i.e., forming/updating beliefs and (2) using the learned priors to perform Bayesian inference.

In chapter 3, I introduced the NEF, a framework that I used in this thesis to build spiking neural models. The NEF constrains the kinds of computations we can use in our models by forcing us to use the basic operations that the neurons can compute. Additionally, when building neural models constrained by the implementation level details, one doesn't get an exact implementation of the algorithm that one may specify at the computational level. Instead the neurons approximate that algorithm and the approximation depends on the neural properties as well as the functions being computed. This is evident from chapter 4 where I showed how an algorithm at the computational level can be implemented at the algorithmic level constrained by implementation level details. I used dimensionality reduction for representing distributions in an efficient way, in order to comply with the limited neural resources in the brain. Furthermore, in chapter 5, we saw from the results that the model performance is sub-optimal, i.e., not Bayes' optimal due to the approximate

computations by neurons, but still matches the human predictions better than an optimal Bayesian model. The low-level constraints also helped to define the architecture of the neural models presented in both of these chapters. To summarize, I have illustrated the importance of integrating different levels of analysis through the process of building the models presented in this thesis.

In chapter 4, I presented a neural model for learning priors from life experience. The assumptions underlying the model were stated and its performance was evaluated by comparing the learned prior to the true prior for a life span inference task. This learned prior was then used in the complete neural model (figure 5.3) presented in chapter 5 to perform Bayesian inference. Through the process of building these models, I showed (1) how probability distributions can be represented in neural circuits using an efficient coding scheme (sections 3.3 and 4.4.4), and (2) how these neural representations can be used in meaningful ways. For instance I showed how the neural representations of an initial prior are improved over time by learning from new incoming observations (chapter 4). I also showed how the neural representations of two different distributions can be combined using an element-wise product to perform Bayesian inference (chapter 5). Finally, I reproduced behavioral results for a life span inference task using the complete neural model that is able to learn prior beliefs and use them for making predictions. The results from the neural model provided a good match to the behavioral data (section 4.4.5). This provides some evidence in support of the hypothesis that human brains represent low-dimensional approximations of high-dimensional distributions fit using neural tuning curves. It also indicates that the computations in the human brain are sub-optimal as discussed in section 6.6, and provides some evidence in support of the neural mechanisms proposed, though more experimental work is needed to validate them as discussed in sections 6.1 and 6.2.

References

Anastasio, T. J., Patton, P. E., and Belkacem-Boussaid, K. (2000). Using bayes' rule to model multisensory enhancement in the superior colliculus. *Neural Computation*, 12(5):1165–1187.

Anderson, C. H. and Van Essen, D. C. (1994). Neurobiological computational systems. *Computational intelligence imitating life*, 213222.

Barlow, H. (1969). Pattern recognition and the responses of sensory neurons. *Annals of the New York Academy of Sciences*, 156(1):872–881.

Barlow, H. and Levick, W. (1969). Three factors limiting the reliable detection of light by retinal ganglion cells of the cat. *The Journal of Physiology*, 200(1):1–24.

Bekolay, T., Bergstra, J., Hunsberger, E., DeWolf, T., Stewart, T. C., Rasmussen, D., Choo, X., Voelker, A., and Eliasmith, C. (2014). Nengo: a Python tool for building large-scale functional brain models. *Frontiers in neuroinformatics*, 7:48.

Berkes, P., Orbán, G., Lengyel, M., and Fiser, J. (2011). Spontaneous cortical activity reveals hallmarks of an optimal internal model of the environment. *Science*, 331(6013):83–87.

Bogacz, R., Brown, E., Moehlis, J., Holmes, P., and Cohen, J. D. (2006). The physics of optimal decision making: a formal analysis of models of performance in two-alternative forced-choice tasks. *Psychological review*, 113(4):700.

Brody, S. and Lapata, M. (2009). Bayesian word sense induction. In *Proceedings of the 12th Conference of the European Chapter of the Association for Computational Linguistics*, pages 103–111. Association for Computational Linguistics.

Chater, N., Oaksford, M., Hahn, U., and Heit, E. (2010). Bayesian models of cognition. *Wiley Interdisciplinary Reviews: Cognitive Science*, 1(6):811–823.

Choo, X. (2010). The ordinal serial encoding model: Serial memory in spiking neurons. Master's thesis, University of Waterloo, Waterloo, ON.

Christoff, K., Prabhakaran, V., Dorfman, J., Zhao, Z., Kroger, J. K., Holyoak, K. J., and Gabrieli, J. D. (2001). Rostrolateral prefrontal cortex involvement in relational integration during reasoning. *Neuroimage*, 14(5):1136–1149.

Cichocki, A. (1992). Neural network for singular value decomposition. *Electronics Letters*, 28(8):784–786.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum likelihood from incomplete data via the EM algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38.

Donoso, M., Collins, A. G., and Koechlin, E. (2014). Foundations of human reasoning in the prefrontal cortex. *Science*, 344(6191):1481–1486.

Doya, K. (2007). *Bayesian brain: Probabilistic approaches to neural coding*. MIT press.

Eliasmith, C. (2005). A unified approach to building and controlling spiking attractor networks. *Neural computation*, 17(6):1276–1314.

Eliasmith, C. (2013). *How to build a brain: A neural architecture for biological cognition*. Oxford University Press.

Eliasmith, C. and Anderson, C. H. (2003). *Neural engineering: Computation, representation, and dynamics in neurobiological systems*. MIT press.

Eliasmith, C. et al. (2012). A large-scale model of the functioning brain (vol 338, pg 1202, 2012). *Science*, 338(6113):1420–1420.

Eliasmith, C. and Martens, J. (2011). Normalization for probabilistic inference with neurons. *Biological cybernetics*, 104(4):251–262.

Fetsch, C. R., Pouget, A., DeAngelis, G. C., and Angelaki, D. E. (2012). Neural correlates of reliability-based cue weighting during multisensory integration. *Nature neuroscience*, 15(1):146.

Fiser, J., Berkes, P., Orbán, G., and Lengyel, M. (2010). Statistically optimal perception and learning: from behavior to neural representations. *Trends in cognitive sciences*, 14(3):119–130.

Friedman, N., Linial, M., Nachman, I., and Pe'er, D. (2000). Using bayesian networks to analyze expression data. *Journal of computational biology*, 7(3-4):601–620.

Furman, M. and Wang, X.-J. (2008). Similarity effect and optimal control of multiple-choice decision making. *Neuron*, 60(6):1153–1168.

Goldwater, S. and Griffiths, T. (2007). A fully bayesian approach to unsupervised part-of-speech tagging. In *Proceedings of the 45th annual meeting of the association of computational linguistics*, pages 744–751.

Gopnik, A., Glymour, C., Sobel, D. M., Schulz, L. E., Kushnir, T., and Danks, D. (2004). A theory of causal learning in children: causal maps and bayes nets. *Psychological review*, 111(1):3.

Gosmann, J. (2015). *Precise multiplications with the NEF*. Technical Report: http://dx.doi.org/10.5281/zenodo.35680, University of Waterloo, Waterloo, Ontario, Canada.

Gosmann, J., Voelker, A. R., and Eliasmith, C. (2017). A spiking independent accumulator model for winner-take-all computation. In *Proceedings of the 39th Annual Conference of the Cognitive Science Society*, London, UK. Cognitive Science Society.

Griffiths, T. L., Chater, N., Norris, D., and Pouget, A. (2012). How the Bayesians got their beliefs (and what those beliefs actually are): comment on bowers and davis (2012).

Griffiths, T. L. and Kalish, M. L. (2005). A bayesian view of language evolution by iterated learning. In *Proceedings of the Annual Meeting of the Cognitive Science Society*, volume 27.

Griffiths, T. L., Kemp, C., and Tenenbaum, J. B. (2008). Bayesian models of cognition.

Griffiths, T. L. and Tenenbaum, J. B. (2006). Optimal predictions in everyday cognition. *Psychological science*, 17(9):767–773.

Hoyer, P. O. and Hyvärinen, A. (2003). Interpreting neural response variability as monte

carlo sampling of the posterior. In *Advances in neural information processing systems*, pages 293–300.

Itti, L., Koch, C., and Niebur, E. (1998). A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 20(11):1254–1259.

Izard, V., Sann, C., Spelke, E. S., and Streri, A. (2009). Newborn infants perceive abstract numbers. *Proceedings of the National Academy of Sciences*, 106(25):10382–10385.

Jacobs, R. A. and Kruschke, J. K. (2011). Bayesian learning theory applied to human cognition. *Wiley Interdisciplinary Reviews: Cognitive Science*, 2(1):8–21.

Jazayeri, M. and Movshon, J. A. (2006). Optimal representation of sensory information by neural populations. *Nature neuroscience*, 9(5):690.

Koch, C. and Ullman, S. (1987). Shifts in selective visual attention: towards the underlying neural circuitry. In *Matters of intelligence*, pages 115–141. Springer.

Kruschke, J. (2014). *Doing Bayesian data analysis: A tutorial with R, JAGS, and Stan*. Academic Press.

Kuss, M., Jäkel, F., and Wichmann, F. A. (2005). Bayesian inference for psychometric functions. *Journal of Vision*, 5(5):8–8.

Lee, D. K., Itti, L., Koch, C., and Braun, J. (1999). Attention activates winner-take-all competition among visual filters. *Nature neuroscience*, 2(4):375.

Levy, R. and Mislevy, R. J. (2016). *Bayesian psychometric modeling*. CRC Press.

Liang, P. and Klein, D. (2009). Online em for unsupervised models. In *Proceedings of human language technologies: The 2009 annual conference of the North American chapter of association for computational linguistics*, pages 611–619. Association for Computational Linguistics.

Lu, H., Yuille, A. L., Liljeholm, M., Cheng, P. W., and Holyoak, K. J. (2008). Bayesian generic priors for causal learning. *Psychological review*, 115(4):955.

Ma, W. J., Beck, J. M., Latham, P. E., and Pouget, A. (2006). Bayesian inference with probabilistic population codes. *Nature neuroscience*, 9(11):1432–1438.

Ma, W. J. and Jazayeri, M. (2014). Neural coding of uncertainty and probability. *Annual review of neuroscience*, 37:205–220.

Mamassian, P., Landy, M., and Maloney, L. T. (2002). Bayesian modelling of visual perception. *Probabilistic models of the brain*, pages 13–36.

Marr, D. (1982). Vision. san francisco: W. h. *H. Freeman*.

Mongin, P. (1997). Expected utility theory. *Handbook of economic methodology*, 342350.

Moreno-Bote, R., Knill, D. C., and Pouget, A. (2011). Bayesian sampling in visual perception. *Proceedings of the National Academy of Sciences*, 108(30):12491–12496.

Narain, D., Remington, E. D., De Zeeuw, C. I., and Jazayeri, M. (2018). A cerebellar mechanism for learning prior distributions of time intervals. *Nature communications*, 9(1):469.

Oja, E. (1982). Simplified neuron model as a principal component analyzer. *Journal of mathematical biology*, 15(3):267–273.

O'Reilly, R. C. (1998). Six principles for biologically based computational models of cortical cognition. *Trends in cognitive sciences*, 2(11):455–462.

Penny, W. D., Friston, K. J., Ashburner, J. T., Kiebel, S. J., and Nichols, T. E. (2011). *Statistical parametric mapping: the analysis of functional brain images*. Elsevier.

Poggio, T. (2012). The levels of understanding framework, revised. *Perception*, 41(9):1017–1023.

Sahani, M. and Dayan, P. (2003). Doubly distributional population codes: simultaneous representation of uncertainty and multiplicity. *Neural Computation*, 15(10):2255–2279.

Schwartz, A. B., Kettner, R. E., and Georgopoulos, A. P. (1988). Primate motor cortex and free arm movements to visual targets in three-dimensional space. I. Relations between single cell discharge and direction of movement. *The Journal of Neuroscience*, 8(8):2913–2927.

Segal, U. (1987). The ellsberg paradox and risk aversion: An anticipated utility approach. *International Economic Review*, pages 175–202.

Sharma, S., Voelker, A. R., and Eliasmith, C. (2017). A spiking neural bayesian model of life span inference. In *Proceedings of the 39th Annual Conference of the Cognitive Science Society*, London, UK. Cognitive Science Society.

Shepard, R. N. (1987). Toward a universal law of generalization for psychological science. *Science*, 237(4820):1317–1323.

Shi, L. and Griffiths, T. L. (2009). Neural implementation of hierarchical bayesian inference by importance sampling. In *Advances in neural information processing systems*, pages 1669–1677.

Singh, R. and Eliasmith, C. (2006). Higher-dimensional neurons explain the tuning and dynamics of working memory cells. *Journal of Neuroscience*, 26(14):3667–3678.

Smith, P. L. and Ratcliff, R. (2004). Psychology and neurobiology of simple decisions. *Trends in neurosciences*, 27(3):161–168.

Spelke, E. S. and Kinzler, K. D. (2007). Core knowledge. *Developmental science*, 10(1):89–96.

Stephan, K. E., Penny, W. D., Daunizeau, J., Moran, R. J., and Friston, K. J. (2009). Bayesian model selection for group studies. *Neuroimage*, 46(4):1004–1017.

Teh, Y. W. (2006). A hierarchical bayesian language model based on pitman-yor processes. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 985–992. Association for Computational Linguistics.

Tenenbaum, J. B., Griffiths, T. L., and Kemp, C. (2006). Theory-based bayesian models of inductive learning and reasoning. *Trends in cognitive sciences*, 10(7):309–318.

Tolhurst, D. J., Movshon, J. A., and Dean, A. F. (1983). The statistical reliability of signals in single neurons in cat and monkey visual cortex. *Vision research*, 23(8):775–785.

Tripp, B. (2009). A search for principles of basal ganglia function. Phd thesis, University of Waterloo.

Trujillo-Barreto, N. J., Aubert-Vázquez, E., and Valdés-Sosa, P. A. (2004). Bayesian model averaging in eeg/meg imaging. *NeuroImage*, 21(4):1300–1319.

Usher, M. and McClelland, J. L. (2001). The time course of perceptual choice: the leaky, competing accumulator model. *Psychological review*, 108(3):550.

Waldmann, M. R. and Martignon, L. (1998). A bayesian network model of causal learning. In *Proceedings of the twentieth annual conference of the Cognitive Science Society*, pages 1102–1107.

Wolpert, D. M. and Landy, M. S. (2012). Motor control is decision-making. *Current opinion in neurobiology*, 22(6):996–1003.

Xu, F. and Tenenbaum, J. B. (2007). Word learning as Bayesian inference. *Psychological review*, 114(2):245.

Zemel, R. S., Dayan, P., and Pouget, A. (1998). Probabilistic interpretation of population codes. *Neural computation*, 10(2):403–430.